IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

TRANSMETA CORPORATION,	)	
	)	
Plaintiff,	)	
	)	
	)	
v.	)	C.A. No. 06-633 (GMS)
	)	
INTEL CORPORATION,	)	
	)	
Defendant.	)	

TRANSMETA CORPORATION'S OPENING CLAIM CONSTRUCTION BRIEF

MORRIS, NICHOLS, ARSHT & TUNNELL LLP
Jack B. Blumenfeld (#1014)
Karen Jacobs Louden (#2881)
Richard J. Bauer (#4828)
klouden@mnat.com
1201 N. Market Street
Wilmington, DE 19899
(302) 658-9200
Attorneys for Plaintiff Transmeta Corporation

OF COUNSEL:
Robert C. Morgan
ROPES & GRAY LLP
1211 Avenue of the Americas
New York, NY 10036-8704
(212) 596-9000

Norman H. Beamer
Sasha G. Rao
Gabrielle Higgins
ROPES & GRAY LLP
525 University Avenue
Palo Alto, CA 94301
(650) 617-4000

John O'Hara Horsley
TRANSMETA CORPORATION
3990 Freedom Circle
Santa Clara, CA 95054

October 19, 2007

TABLE OF CONTENTS

TABLE OF AUTHORITIES

STATUTES

TABLE OF ABBREVIATIONS

The abbreviations in this memorandum use the following format:

"A__"	JA-A page numbers in the Joint Appendix for the Transmeta patents
"B__"	JA-B page numbers in the Joint Appendix for the Transmeta patents
"D__"	JA-D page numbers in the Joint Appendix for the Intel patents
"A__, 1:2-4"	Page number in the Joint Appendix, followed by patent column and line numbers
"Term __"	The claim terms as numbered in the Joint Claim Construction Chart, D.I. 92
'061 patent	Transmeta U.S. Patent No. 7,100,061 (A1-13)
'687 patent	Transmeta U.S. Patent No. 5,493,687 (A14-33)
'986 patent	Transmeta U.S. Patent No. 5,838,986 (A34-55)
'449 patent	Transmeta U.S. Patent No. 6,044,449 (A56-76)
'624 patent	Transmeta U.S. Patent No. 5,737,624 (A77-97)
'526 patent	Transmeta U.S. Patent No. 5,974,526 (98-119)
'433 patent	Transmeta U.S. Patent No. 6,289,433 (A120-41)
'503 patent	Transmeta U.S. Patent No. 5,895,503 (A142-57)
'733 patent	Transmeta U.S. Patent No. 6,226,733 (A158-74)
'668 patent	Transmeta U.S. Patent No. 6,430,668 (A175-88)
'669 patent	Transmeta U.S. Patent No. 6,813,699 (A189-201)
'375 patent	Intel U.S. Patent No. 5,745,375 (D1-10)
'554 patent	Intel U.S. Patent No. 5,617,554 (D11-34)
'605 patent	Intel U.S. Patent No. 5,802,605 (D35-57)
'101 patent	Intel U.S. Patent No. 5,819,101 (D58-86)
'275 patent	Intel U.S. Patent No. 5,881,275 (D87-114)
'634 patent	Intel U.S. Patent No. 6,385,634 (D115-35)
'529 patent	Intel U.S. Patent No. 6,418,529 (D136-58)

I. NATURE AND STAGE OF THE PROCEEDINGS

Transmeta Corp. ("Transmeta") filed this action on October 11, 2006, asserting that Intel Corporation ("Intel") infringes eleven Transmeta patents. Intel has asserted counterclaims, including claims that Transmeta infringes seven Intel patents. Pursuant to the Court's Scheduling Order (D.I. 25), fact discovery is set to close on February 1, 2008. Expert discovery closes on May 9, 2008. A pretrial conference is scheduled for November 3, 2008 and trial is set to begin on December 1, 2008. The parties filed their Joint Claim Construction Chart on October 10, 2007 (D.I. 92). This is Transmeta's opening claim construction brief directed to both the Transmeta and Intel patents.

II. SUMMARY OF THE ARGUMENT

Transmeta's approach to claim construction is consistent. Whether raising potential terms for construction in Transmeta's patents or proposing a construction for terms in Intel's patents, Transmeta bases its construction on the claim language and the intrinsic evidence.

Intel's approach to claim construction is self-servingly inconsistent. When construing its own patents, Intel believes that few terms need construction, and those that are construed are calculated to seek infringement where there is none, or to avoid invalidity. For example, in construing the term "processor" in its own patent, Intel says that no construction is required. Intel asserts that when construing Transmeta's patents, the word "processor" should be limited to only a processor using a specific type of instruction set.

III. STATEMENT OF FACTS

A. The Transmeta Patents

The eleven Transmeta patents in suit are in four separate "families" of patents. The first family consists of one patent, U.S. Patent No. 7,100,061 ("the '061 patent"), and relates to adaptive power control. The "Multiple Typed Register" patents consist of U.S. Patents Nos.

5,493,687; 5,838,986 and 6,044,449. The "Register Renaming" patents consist of U.S. Patents Nos. 5,737,624; 5,974,526 and 6,289,433. The "Speculative Address Translation" patents consist of U.S. Patents Nos. 5,895,503; 6,226,733; 6,430,668 and 6,813,699. An overview of each Transmeta patent family is included in the discussion of the proper construction of the disputed terms below.

B. The Intel Patents

The seven Intel patents in suit are in five separate "families" of patents. The first family consists of one patent, U.S. Patent No. 5,745,375 (the "Power Usage" patent). The "Address Translation" patents consist of U.S. Patent Nos. 5,617,554 and 5,802,605. The "Pack/Unpack" patents consist of U.S. Patent Nos. 5,881,275 and 5,819,101. The fourth and fifth families each consist of one patent: U.S. Patent No. 6,385,634 (the "Multiply-Add" patent) and U.S. Patent No. 6,418,529 (the "Intra-Add" patent). An overview of each Intel patent family is included in the discussion of the proper construction of the disputed terms below.

IV. THE PROPER CONSTRUCTION OF THE DISPUTED TERMS

A. The '061 Patent

The '061 patent in suit, entitled "Adaptive Power Control," describes methods and apparatus for controlling the power consumed by a processor. Software monitors the operating characteristics of the processor. The monitored conditions may include, for example, present frequency and voltage of operation, temperature of operation, or the amount of time that the processor spends in one or more idle states (*e.g.*, halt or sleep). The processor's power consumption is a function of the operating voltage and frequency of the processor. Based on its monitored conditions or operating characteristics, the processor determines an appropriate frequency and appropriate voltage of operation and changes the power consumption by changing the frequency and voltage of operation to the determined values. In this manner, the processor

can reduce power consumption, thereby increasing the operating battery life of portable computers or similar devices. To further increase the efficiency of the processor, the processor continues to execute instructions while changing the voltage.

1. "determining a maximum allowable power consumption level..."

Claim Language	Proposed Constructions
determining a maximum allowable power consumption level from an operating condition of the processor, said computer processor determining a maximum frequency which provides power not greater than the allowable power consumption level, said computer processor determining a minimum voltage which allows operation at the maximum frequency determined (claim 1)	**Transmeta** - based on an operating condition of the processor, the computer processor determines a maximum allowable power consumption level by determining a corresponding maximum frequency and a minimum voltage which allows operation at the maximum frequency.
	Intel - after determining a maximum allowable power consumption level from an operating condition of the processor, the computer processor determines, in a separate step, a maximum frequency which provides power not greater than the determined allowable power consumption level, and the computer processor determines, in another separate step, a minimum voltage which allows operation at the determined maximum frequency.

Transmeta's proposed construction is supported by the claim language itself, which states that the maximum allowable power consumption level is determined by "determining a maximum frequency which provides power not greater than the allowable power consumption level... [and a] minimum voltage which allows operation at the maximum frequency determined." (A10, claim 1). It also is consistent with the specification, which discloses that power usage is determined by the voltage and the frequency expressed in equation form as $P=CV^2f$ (A7, 1:42-56), as well as with the description of the preferred embodiment (A4, Fig. 2; A9, 5:15-67).

Intel's proposed construction is contrary to the claim language and separates determining a maximum allowable power consumption level from the very steps which make that determination. It would create separate steps, when they are not separate, and would read the preferred embodiment out of the claim.

2. "determining a frequency and a voltage" / "determining a voltage-frequency pair"

Claim Language	Proposed Constructions
"determining a . . . frequency. . . [and a] voltage" (claim 1)	**Transmeta** - No construction necessary – plain and ordinary meaning.
"determining a frequency and a voltage" (claims 15, 23, 30) "determining a voltage-frequency pair" (claim 39)	**Intel** - determine a frequency and a voltage based at least on analyzing commands to be executed by the processor.

These are well understood terms that do not need construction. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005). Intel does not actually construe these terms, but improperly seeks to add the limitation "*based at least on analyzing commands to be executed by the processor*." Intel appears to be adding this limitation based on its position that the specification never discloses increasing the frequency and voltage of operation without "analyzing commands to be executed by the processor." This ignores Figure 2, which shows that the control software monitors operating conditions (such as present frequency and voltage of operation, temperature of operation, or the amount of time that the processor spends in one or more idle states), without any analysis of commands to be executed, and determines whether those conditions indicate that the frequency and voltage of operation should be changed, and then increases or decreases the frequency and voltage accordingly. Intel's attempt to narrow the claim language to exclude the Figure 2 embodiment should be rejected.

3. "clock frequency generator"

Claim Language	Proposed Constructions
"clock frequency generator" (claims 8, 56)	**Transmeta** - No construction necessary – plain and ordinary meaning.
	Intel - a unit that provides individual clock frequencies for each of a plurality of components including a processing unit of the processor, the system memory, and the system bus.

This term requires no construction, and should be given its plain and ordinary meaning. Intel's proposed construction would insert the limitations that the clock frequency generator "*provide individual clock frequencies for each of a plurality of components including a processing unit of the processor, the system memory, and the system bus.*" Moreover, the doctrine of claim differentiation prevents the term "clock frequency generator" from being construed to require these limitations in claim 56. Claim 58 depends from claim 56 and explicitly provides that the clock frequency generator "is further operable to concurrently generate frequencies for a plurality of functional units of the computing device." (A12-13). To define clock frequency generator as proposed by Intel would impermissibly eliminate any distinction between claims 56 and 58 of the '061 patent.

Claim 8 states that the clock frequency generator provides "one of a plurality of selectable output clock frequencies to the processing unit." (A10). Accordingly, the claim itself specifies what the clock frequency generator provides, and it is not what Intel would add to the clock frequency generator element.

4. "means for detecting. . . and causing"

Claim Language	Proposed Constructions
means for detecting the values indicative of operating conditions of the central processor and causing the power supply and clock frequency generator to furnish an output clock frequency and voltage level for the central processor and to generate concurrently frequencies which are selected for optimum operation of a plurality of functional units of the computing device (claim 8)	**Transmeta** - The structure is control software and a set of registers in the processor – such as the clock divider register 22 – in which are stored a multiplier and dividers computed by the processor (or determined via table lookup) based on operating conditions of the processor.
	Intel - The structure is control software executing on the processor and the cooperating hardware on the processor.

The parties agree that this claim element is a means-plus-function limitation that must be construed according to 35 U.S.C. § 112, ¶ 6, and that the claimed function is as recited. They disagree as to the structure that corresponds to the specified function.

The specification discloses that the structure used for performing the specified function is control software and a set of registers in the processor – such as the clock divider register 22 – in which a multiplier and dividers computed by the processor (or determined via table lookup) are stored. (A3, Fig. 1, master control unit 18; A5, Fig. 3, clock divider register 22; A8, 4:29-54; A9, 5:21-67).

Intel's proposed construction, "control software executing on the processor and the cooperating hardware on the processor," is impermissibly vague and fails to identify any particular structure disclosed in the specification.

5. "a state of said computer processor"

Claim Language	Proposed Constructions
"a state of said computer processor" (claim 30)	**Transmeta** - condition of the computer processor such as activeness or idleness.
	Intel - the activeness or idleness of the computer processor.

Transmeta's proposed claim construction is consistent with the claim language and the specification. (A12). The '061 patent's claims and specification provide examples of various states of a computer processor, including halt, sleep and frequency states of operation. (A12, claims 31-33; A9, 5:21-45; A10, 7:49-53). Intel's construction impermissibly narrows "a state of said computer processor" to activeness or idleness, ignoring the other states disclosed in the patent.

6. "said computer processor determining a frequency and a voltage" / "a computer processor determining a voltage-frequency pair"

Claim Language	Proposed Constructions
"said computer processor	**Transmeta** - No construction necessary – plain and ordinary

determining a frequency and a voltage" (claims 23, 30)	meaning.
	Intel - the computer processor itself, not the operating system, determines a frequency and a voltage.
"a computer processor determining a voltage-frequency pair" (claim 39)	**Transmeta** - No construction necessary – plain and ordinary meaning.
	Intel – a computer processor itself, not the operating system, determines a pair of voltage and frequency values.

These terms require no construction, and should be given their plain and ordinary meaning. Intel does not actually construe these terms, but improperly seeks to add the limitation "*not the operating system.*" In the joint claim chart, Intel cites to applicants' arguments made during prosecution that the Horden patent:

> fails to teach or suggest the claimed limitation of a computer processor determining the frequency and voltage, which is vastly different from an operating system performing these tasks.

(A290).[1] The Examiner explicitly rejected applicants' arguments, however, stating:

> [In Horden] [t]he operating system runs on the processor. Thus the processor determines the frequency and voltage at which to operate the processor. ... In Horden the control software is the operating system.

(A306). Thereafter, the applicants disavowed their arguments by acquiescing to the Examiner's position and amended the claims in other respects. (A329). Based on these amendments and the remarks discussing and explaining them, the Examiner allowed the issued claims. (A341; B1-3). Therefore, applicants did not rely on the Horden operating system arguments for any purpose, the Examiner explained that the claim language includes the operating system running on the processor, and there is therefore no basis for Intel's proposed limitation. [2]

[1] Applicant's prosecution statements were factually incorrect and inconsistent with the application's disclosure. Under *Intervet Am. v. Kee-Vet Labs.*, 887 F.2d 1050, 1054 (Fed. Cir. 1989), such factually incorrect statements cannot result in a disclaimer.

[2] Intel's proposal should also be rejected because contrary to its position in this litigation, Intel has argued in the '061 reexamination that the claim language includes the processor itself *and the operating system running on the processor*. (B5).

7. "operating conditions of the central processor" / "operating conditions internal to said computer processor" / "internal conditions of a computer processor"

Claim Language	Proposed Constructions
"operating conditions of the central processor" (claim 8) "operating conditions internal to said computer processor" (claim 15) "internal conditions of a computer processor" (claim 39)	**Transmeta** - No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means a plurality of types of operating conditions that are internal to the computer processor.
	Intel - a plurality of types of operating conditions, excluding core utilization, that are internal to the computer processor.

These terms require no construction, and should be given their plain and ordinary meaning. Intel again does not actually construe these terms, but improperly seeks to add the limitation "*excluding core utilization.*" In the joint claim chart, Intel again cites to the Horden operating system arguments that applicants made during prosecution. (A253). As discussed above, these arguments were explicitly rejected by the Examiner and disavowed by applicants. In rejecting applicants' arguments, the Examiner construed the '061 claims, stating: "The core utilization is an internal condition. ... Core utilization includes idle time." (A267). The specification and claims also provide that idle time is an operating condition internal to the computer processor. (A11-12, claims 23, 44-47; A9, 5:21-45). Thus, for the same reasons as discussed in the previous section, there is no basis for adding Intel's proposed limitation.[3]

[3] Intel's construction should also be rejected because contrary to its position in this litigation, Intel has argued in the '061 reexamination that core utilization (processor load) *is* a condition of the processor. (B6).

B. The Multiple Typed Register Patents[4]

Computer processors typically include storage locations on the processor called registers. (A45, 1:53-57). Registers provide temporary storage that the processors can access more quickly than memory located outside of the processor, such as the main memory. There are a variety of types of registers, including, for example, address registers and general purpose registers. (A45, 1:53-57). One type of register that is common to computer processors is a data register. (A45, 1:53-59).

Data registers are normally used to hold either integer or floating point data. (A45, 2:42-44). Integer data is "numeric data which is used to represent a positive or negative whole number or zero," *e.g.*, "2" or "74." (D.I. 92, Tab A, p. 129). Floating point data, on the other hand, is numeric data that is "represented by a positive or negative sign, the digits in the number, and an exponent, specifying the magnitude of the number, *e.g.*, "-3.0×10^5." (*Id*).

The Multiple Typed Register patents emphasize a shortcoming of prior art as having only one set of registers for a single data type. (A24, 2:41-49). Prior art processors often included one set of registers for holding integer data and a separate set of registers for holding floating point data, "with each set being limited to its respective data type." (*See* A24, 2:60-65). This arrangement, however, resulted in inefficiencies. (A45, 2:66). Because "user applications frequently involve exclusively integer operations, and perform no floating point operations whatsoever," the separate set of floating point registers often "remain idle during the entire execution." (A25, 3:2-7).

[4] The Multiple Typed Register patents are the '687, '986 and '449 patents. Although there are three separate patents in the family, the specifications are identical; only the claims differ. Therefore, the references herein are made to the '687 patent, with the understanding that they apply equally to the other two patents.

Recognizing this inefficiency, the Multiple Typed Register patents provide for "an architecture which allows multiple register sets within a given data type," teaching that it is "desirable for a microprocessor's floating point registers to be usable as integer registers." (A25, 3:54-60). The patent accomplishes this with "a plurality of integer register sets" whereby "[o]ne of the register sets is also usable as a floating point register set." (A25, 4:55-63). The specification further explains that "Register Set A" includes "integer registers 24 (RA[31:0]), each of which is adapted to hold an integer value datum," and "Register Set FB" includes "retypable register set FB 20 [which] may be thought of as including floating point registers 48 (RF[31:0]), and/or integer registers 50 (RB[31:0])." (A26, 5:63-66, A27, 7:26-30). The two register sets are depicted in Figure 1 (highlighted in yellow). (A15, Fig. 1).

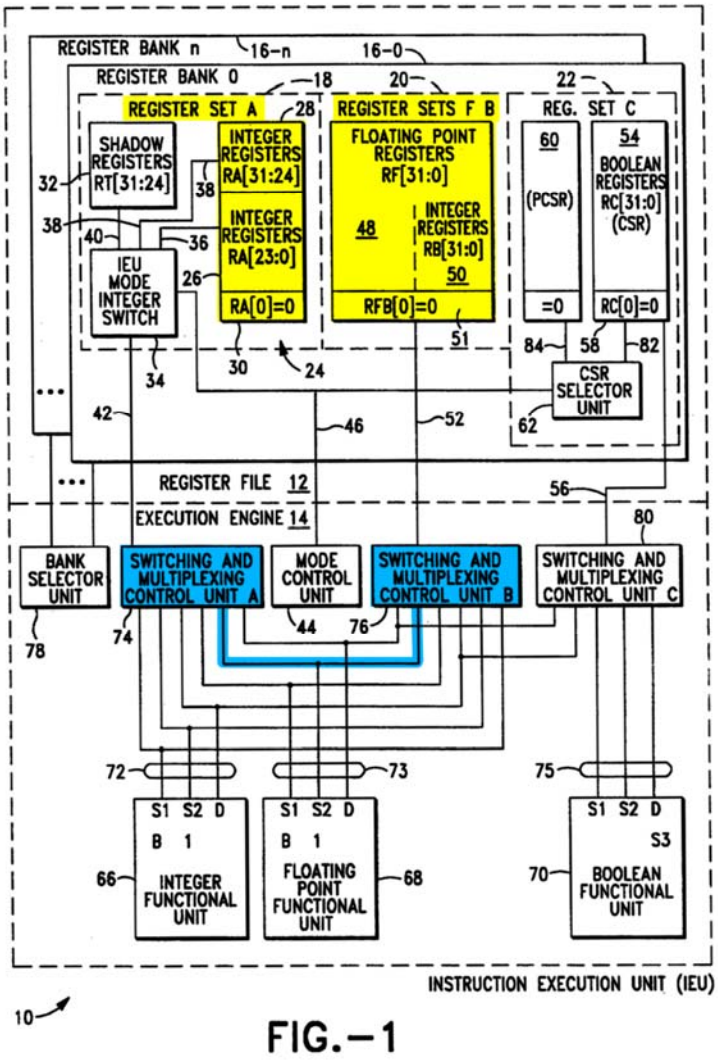

FIG.—1

The Multiple Typed Register patents also disclose the use of Switching and Multiplexing Control Units (SMC units) (highlighted in blue above), which determine whether to read or write an operand value in either the integer register set or the floating point register set. (A29, 11:8-18; *see* A15, Fig. 1). The SMC units are responsive to bits in an instruction that specify which

register is to be accessed. (A25, 4:58-60, A28, 10:42-48, A28-29, 10:64-11:4, 11:12-14, 11:56-59). Importantly, because the SMC units are connected (as shown in blue above), they allow operands to be moved from one register set to the other. That is, if an operand is read from one register set, the connectivity between the register sets allows the operand to be written to the opposite register set. This type of "register-to-register" instruction is "of particular interest" to these patents. (A28, 9:16-17).

Multiplexers, and an integer functional unit, allow connection and flow of data between and to either register set (A25, 4:57-60; A28, 10:64-11:4). In the preferred embodiment, register sets are coupled to receive data from, and send data to, each other on buses 114, 118, 116 and 120 of Figs. 2, 2A, 3, 3A. (A31, 15:21-22, 15:40-42, 15:52-54, 16:1-2, 16:37-39, 16:48-49, 16:56-59; A32, 17:1-5, 17:16-21, 17:39-41).

There are sixteen disputed terms in the Multiple Typed Register patents. Transmeta's proposed constructions for all of the disputed terms are provided in D.I. 92, Tab A, pp. 129-143.

1. "central processing unit" / "processor"

Claim Language	Proposed Constructions
"central processing unit" / "processor"	**Transmeta** – No construction necessary – plain and ordinary meaning.
('687 claim 1; '986 claim 1; '449 claim 1)	**Intel** – a processor in a reduced instruction set computer.

The terms "central processing unit" and "processor" appear in the preamble of the asserted claims and need no construction. (*See, e.g.*, A32, 18:45-48; A76, 18:5). These terms are readily understood by a jury and should be given their ordinary meaning.

Intel seeks to limit the claims of the Multiple Typed Register patents to microprocessors having a "Reduced Instruction Set" (RISC) by construing the common terms "central processing unit" and "processor" to mean "a processor in a reduced instruction set computer." (D.I. 92, Tab

A, p. 129). Intel's proposed construction asks that the Court improperly import a limitation from the specification into the preamble by redefining a term that has an ordinary meaning. *See Comark Communs., Inc. v. Harris Corp.*, 156 F.3d 1182, 1186 (Fed. Cir. 1998).

Although some of the ***unasserted*** claims specify a RISC processor in the preamble (*see, e.g.*, A54, claim 13), ***none of the asserted claims recite RISC or any other type of instruction set***. (*See, e.g.*, A53, claim 1). Moreover, processors with a complex instruction set (CISC) also have integer registers and floating point registers, and can use the claimed inventions of the Multiple Typed Register patents to improve their performance. (B1010).

2. "first registers each for holding the integer data" and "second registers each for holding the integer data and for holding floating point data"

Claim Language	Proposed Constructions
"first registers each for holding the integer data" ('687 claim 1; '986 claim 1; '449 claim 1)	**Transmeta** – storage locations identifiable by instructions and capable of storing only integer data.
	Intel – storage locations identifiable by instructions and capable of storing at least integer data.
"second registers each for holding the integer data and for holding floating point data" ('687 claim 1; '986 claim 1; '449 claim 1)	**Transmeta** – each second register can hold integer data and, alternatively, can hold floating point data.
	Intel – storage locations identifiable by instructions and capable of storing at least integer data and floating point data.

The "first registers" and "second registers" in the asserted claims refer to the integer and floating point registers taught by the patent (*See, e.g.*, A32, claim 1). Transmeta's constructions properly define the role of each register set recited in the claims. The first register set is "capable of storing ***only*** integer data," and the second register set "can hold integer data and, ***alternatively***, can hold floating point data." (D.I. 92, Tab A, p. 129). Intel's constructions improperly include the phrase "at least," which would eliminate any distinction between the two register sets by allowing either register set to hold either type of data. (D.I. 92, Tab A, p. 129). This ignores the

context from which the Multiple Typed Register patent arose. The prior art inefficiently devoted one register set to integer data and another register set to floating point data. In contrast to the prior art, the patents allow the floating point registers to also be used for integer data.

"first registers." The claims state that the first register set is for holding "***the*** integer data." (*e.g.*, A32, claim 1). The specification teaches that the first register set (or "Register Set A") includes "integer registers 24 (RA[31:0]), each of which is ***adapted to hold an integer value*** datum." (A26, 5:63-66) (emphasis added). There is no suggestion that the integer register set can hold any other data type. Indeed, the patent labels Register Set A in Figure 1 as "Integer Registers RA." (A15, Fig. 1; A26, 5:65-66). And during prosecution the patentee explained that integer registers, which are 32 bits long, are ineffective for holding 80-bit floating point data because it would result in a loss of precision. (A363).

Intel proposes that the first register set can hold "***at least*** integer data." (D.I. 92, Tab A, p. 129). This construction would improperly allow the integer register set to also hold floating point data, which eliminates the claimed distinction between the two claimed register sets. It is also inconsistent with the language of the claim, which requires the register to hold "the integer data." (*See, e.g.*, A32, claim 1).

"second registers." The claims state that the second registers are "each for holding the integer data and for holding floating point data." (*See, e.g.*, A32, claim 1). This refers to the fact that the second registers are capable of holding both types of data. As the specification further explains, at any point in time, "[e]ach individual register in the register set RFB[] may hold

either a floating point value ***or*** an integer value." (A27, 7:42-43). That is, each register in the set can hold integer data or floating point data, but not both at the same time.[5]

During prosecution of the '687 patent, applicants emphasized the distinction between the two register sets by arguing that the "claimed invention recites an integer register set for storing integer data and a re-typable register set for storing floating point or integer data. Thus, Applicants' claimed invention provides a second source for storing integer operands and results." (A363).[6] Accordingly, the intrinsic evidence is consistent with the claim requirement that the first register holds only integer data and the second register holds integer data or, alternatively, floating point data.

Intel proposes instead that the second register set can hold "***at least*** integer data and floating point data." (D.I. 92, Tab A, p. 130). But again, there is no suggestion in the claims, specification or file history that the second register can hold any type of data other than what is claimed – *i.e.*, integer or floating point data. The intrinsic evidence emphasizes that the advantage of the invention is that the floating point registers are "usable as integer registers in case the available integer registers are inadequate to optimally hold the necessary amount of integer data." (A353; A25, 3:58-60).

3. "a field" . . . "specifying which of the first and second register sets is to be accessed"

Claim Language	Proposed Constructions
"a field"	**Transmeta** – one or more bit locations in a computer instruction.

[5] Transmeta's understanding is that the parties do not disagree on this point. However, Intel's proposed construction is ambiguous on this point, and therefore Transmeta's construction should be adopted.

[6] Although these argument were made with respect to other claims, they apply with equal force to the issued claims of the '687 patent. In particular, applicants stated that claim 1 of the '687 patent was allowable because it had been amended to recite "a first register set for storing integer values and a second register set for storing floating point and integer values." (A365).

('687 claim 1; '986 claim 1)	**Intel** – a dedicated portion of an instruction having a defined meaning.
"specifying which of the first and second register sets is to be accessed" ('687 claim 1; '986 claim 1)	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means that the field in the instruction indicates which register set is to be accessed.
	Intel – the instruction is modifiable so that it performs the operation utilizing either of the two register sets.

Because the claimed invention includes multiple register types, instructions executed by the processor must indicate which register is to be accessed. The claims include this element by reciting that "a specific instruction ***includes a field specifying which of said first and second register sets is to be accessed*** in response to execution of said specific instruction." (A53, claim 1) (emphasis added).

"**field."** "Field" should be given its ordinary meaning of "one or more bit locations in a computer instruction." A "field" is defined by IEEE as "a set of bit locations in a computer word used to express the address of the operand." (B1003).

Figure 7 of the patent provides an exemplary processor instruction that includes a field specifying the register locations from which operands are to be read from or written to. (A23, Fig. 7; A26, 5:31-33). As shown in the figure, the instruction I[] includes multiple bit locations, *i.e.*, B0, B1 and B2, that indicate which register the instruction reads from and writes to. (A23, Fig. 7). Figures 5 and 6 show that the multiple bits B0, B1 and B2 from the instruction field are read by multiplexers S1 and S2 to determine which register to access. (A21, Fig. 5; A22, Fig. 6; A28, 9:21-23, 9:30-34, 9:36-38, 10:64-11:4; A29, 11:12-14).

Intel proposes that the field should be limited to a "dedicated portion" of the instruction that has a defined meaning. (D.I. 92, Tab A, p. 130). But Figure 7 shows that the field of the instruction specifying which register to access is not located in a single contiguous portion. Rather, it can be spread across multiple bit locations. Limiting the field of the claims to only one

of those portions would exclude the embodiment of Figure 7 from the claims. Moreover, there is no reason to limit the field to a "dedicated" portion of an instruction. The claims do not prohibit the same field from performing other actions. Nor do they prohibit different fields of different instructions from performing the required function.

"**specifying which of the first and second register sets is to be accessed.**" This phrase does not need to be construed and should be given its ordinary meaning. A jury would easily understand that the claim requires the field to specify "which of the first and second register sets is to be accessed." (*See, e.g.*, A32, claim 1). If the Court feels it is necessary to construe this language to assist the jury, Transmeta proposes that it be construed to mean that the field in the instruction indicates which register set is to be accessed.

Intel's construction entirely rewrites and changes the meaning of the claim. The claims require ***a field*** in the instruction ***to specify*** which register set to access. Intel's construction, however, says nothing about the role of the field, or about specifying which register to access. Instead, Intel requires ***an instruction*** to be ***modifiable*** so that it can perform an operation utilizing ***either*** register set. This has no basis and is contrary to the claim language and the specification. The claims call for two separate register sets and a field in the instruction that specifies which set to access. (*See, e.g.*, A32, claim 1).

4. "reading means . . ." and "writing means . . ."[7]

Claim Language	Proposed Constructions
"reading means for	**Transmeta** – The function performed by the claimed "reading

[7] The '449 patent requires an "execution unit" that "reads an operand value from either said first register [or] second register set as specified by said instruction" or "writes a result value to said first register set or said second register set as specified by said instruction." (*See* A76, claim 1). Transmeta's proposed constructions for the "execution unit" terms are consistent with its proposed constructions for the functions of the "reading means" and "writing means." Intel has likewise proposed similar constructions. Therefore, Transmeta does not repeat its arguments for those terms in this brief.

reading an operand value from either the first register set or second register set as specified by the field" ('687 claim 1; '986 claim 1)	means" is reading an operand value from either the first register set or second register set as specified by the field. The disclosed structures that correspond to the function of the claimed reading means is the multiplexing circuitry within the SMC units A and B (for SMC unit A, see those S1/S2 MUXs that participate in the selection between register sets; for SMC Unit B, see those S1/S2 MUXs that participate in the selection between register sets).
	Intel – The claimed function is "reading data in response to an instruction that is modifiable to perform an operation utilizing either of the two register sets for source data." The corresponding structure includes at least the 12 multiplexers labeled S1, S2 in figures 2A and 3A. The multiplexers are controlled by instruction bits B1 (SOURCE 1) and B2 (SOURCE 2).
"writing mean[s] for writing a result value to the first register set or the second register set as specified by the field" ('687 claim 1; '986 claim 1)	**Transmeta** – The function performed by the claimed "writing means" is for writing a result value to the first register set or the second register set as specified by the field, and which is capable of writing a result value to the first register set if the reading means reads an operand value from the second register set, and vice versa. The disclosed structure in the patent specification that correspond to the function of the claimed "writing means" is multiplexing circuitry within the SMC units A and B (for SMC unit A, see MUX circuits 148; for SMC unit B, see MUX circuits 110).
	Intel – The claimed function is "writing data in response to an instruction that is modifiable to perform an operation utilizing either of two register sets for destination data." The corresponding structure includes at least the 4 multiplexers labeled 110- , 110- , 148- , and 148- in figures 2 and 3. The multiplexers are controlled by instruction bit BO (DEST.) of figure 7.

The parties agree that the terms "reading means" and "writing means" recite means-plus-function elements that must be construed according to 35 U.S.C. §112, ¶6.

The Claimed Function of the "reading means." The claims recite the function of the reading means as "reading an operand value from either the first register set or second register set as specified by the field." (*See, e.g.*, A32, claim 1). Transmeta's construction of the claimed function is identical to the claim language. (D.I. 92, Tab A, p. 131).

Intel's construction improperly requires the reading means to read data "in response to an instruction that is modifiable to perform an operation utilizing either of the two register sets for source data." (D.I. 92, Tab A, p. 131). This additional limitation is not called for by the claim

language or by the specification. The claims expressly state that the reading means selects one of the registers "as specified ***by the field***." There is no support for requiring "an instruction that is modifiable."

The Claimed "reading means." Because the reading means must be capable of reading an operand from ***either*** the first register set or second register set, only those multiplexers S1 and S2 that participate in the selection between register sets should be included in the structure. *Micro Chem., Inc. v. Great Plains Chem. Co.*, 194 F.3d 1250, 1258 (Fed. Cir. 1999) (35 U.S.C. §112, ¶6 does not "permit incorporation of structure from the written description beyond that necessary to perform the claimed function.").

Figures 5 and 6 of the patent depict multiplexing circuitry S1 and S2 within SMC units A and B that is responsive to the field and selects between the two register sets. (A31, 16:6-15; A32, 17:49-50; e.g., A21, Fig. 5; A22, Fig. 6). Figures 5 and 6 show multiplexers S1 and S2 receiving bits B1 and B2 and, depending upon the value of the bits, selecting between one of the registers in integer register set (RA[]) or floating point register set (RFB[]). (A21, Fig. 5; A22, Fig. 6). As shown in Figures 5 and 6, not all of the multiplexers S1 and S2 are able to read from ***either*** register set. (A21, Fig. 5; A22, Fig. 6). The top two multiplexers S1 and S2 in Figure 5, for example, are only connected to floating point registers RFB2, RFB0, RFB1 and RFB3. (A32, Fig. 5). These multiplexers should not be included in the claimed "reading means" because they are not capable of reading an operand value from either the floating point register set RFB[] or the integer register set RA[].

Intel's construction should be rejected because it requires all 12 multiplexers labeled S1 and S2 to be part of the structure, including those that are incapable of reading from the integer register set. Intel also adds a requirement that the multiplexers are controlled by instruction bits

B1 (SOURCE 1) and B2 (SOURCE 2). This additional functional limitation is not called for by the claims which require only that the reading means read from a register that is specified by the field. The Federal Circuit has instructed that §112, ¶6 "does not permit limitation of a means-plus-function claim by adopting a function different from that explicitly recited in the claim." *Micro Chemical*, 194 F.3d at 1258.

The Claimed Function of the "writing means." The claim recites that the function of the "writing mean[s]" is "writing a result value to the first register set or the second register set as specified by the field." (*See, e.g.*, A32, claim 1). The patent also indicates that the writing means must be further capable of writing a result value to the first register set if the reading means reads an operand value from the second register set, and vice versa. This is required in order to fully satisfy the expressed object of the invention of the Multiple Typed Register patents of having "an architecture which allows multiple register sets within a given data type." (A25, 3:54-56; *also*, 4:54-62). In order to realize this capability, the patents disclose an architecture that allows the register sets to freely transfer the integer values from one register set to another. (4:57-60; 10:64-11:4). In the preferred embodiment, register sets are coupled exchange data via buses 114, 118, 116 and 120 of Figs. 2, 2A, 3, 3A. (A31, 15:21-22, 15:40-42, 15:52-54, 16:1-2, 16:37-39, 16:48-49, 16:56-59; A32, 17:1-5, 17:16-21, 17:39-41).

As with the "reading means," Intel's proposed construction of "writing means" requires "an instruction that is modifiable to perform an operation utilizing either of the two register sets for source data." (D.I. 92, Tab A, p. 132). This additional limitation is improper for the same reasons discussed above.

The Claimed "writing means." The parties agree that the specification discloses multiplexing circuitry 148 and 110 within the SMC units A and B, which is capable of writing to

the first and second register sets, respectively. (A31, 16:35-38, 15:20-23; *see also* A16, Fig. 2; A18, Fig. 3). This circuitry performs the claimed function of writing to the register that is specified by the instruction field. The circuitry is also capable of writing to the opposite type of register than the register type that is read from. Indeed, the specification points out that "register-to-register" instructions are an "[i]nstruction class of particular interest to this Application." (A28, 9:16-17).

As with the reading means, Intel's construction requires all "4 multiplexers labeled 110- , 110- , 148- , and 148- in figures 2 and 3" to be part of the disclosed structure. (*See* D.I. 92, Tab A, p. 132). Intel also adds a requirement that "the multiplexers are controlled by instruction bit B0 (DEST.) of figure 7." (D.I. 92, Tab A, p. 132). For the same reasons discussed with respect to the "reading means," Intel's construction should be rejected.

5. "read access means" and "write access means"

Claim Language	Proposed Constructions
"read access means, responsive to the data processing system performing a given read operation of a specific data type, for accessing said register set to retrieve data from a given register" ('986 claim 11)[8]	**Transmeta** – The function performed by the claimed "read access means" is accessing a register set to retrieve data from a given register. The disclosed structures that correspond to the function of the claimed reading means is the multiplexing circuitry within the SMC units A and B (for SMC unit A, see MUXs 150; for SMC Unit B, see MUXs 112).
	Intel – The claimed function is "obtaining source data from a register set." The corresponding structure includes at least the 12 multiplexers labeled S1, S2 in figures 2A and 3A. The multiplexers are controlled by instruction bits B1 (SOURCE 1) and B2 (SOURCE 2).
"write access means, responsive to the data processing system performing a given write operation, for accessing said register set to store into a given register"	**Transmeta** – The function performed by the claimed "write access means" is accessing a register set to store into a given register data specified by said write operation. The disclosed structure in the patent specification that correspond to the function of the claimed "writing means" is multiplexing circuitry within the SMC units A and B (for SMC unit A, see MUX circuits 148; for SMC unit B, see MUX circuits 110).

[8] This language appears in claim 6 from which claim 11 depends. (A53-54).

('986 claim 11)	**Intel** – The claimed function is storing result data to the register set. The corresponding structure includes at least the 4 multiplexers labeled 110- , 110- , 148- , and 148- in figures 2 and 3. The multiplexers are controlled by instruction bit (DEST.) of figure 7.

The Claimed Functions of the "read access" and "write access" Means. In contrast to the reading and writing means of claim 1, the "read access means" and "write access means" in claim 11 of the '986 patent do ***not*** need to be capable of reading from or writing to ***either*** register set. Rather, the claim requires that the "access means" provide access to a ***single*** register set containing a ***specific*** data type. The claim states that the "read access means [is] responsive to the data processing system ***performing a given read operation of a specific data type***." (A53, claim 11) (emphasis added). The claimed functions, therefore, are "accessing a register set to": (i) "retrieve data from a given register," in the case of the read access means, and (ii) "store into a given register data specified by said write operation," in the case of the write access means. (D.I. 92, Tab A, pp. 136-37).

The Claimed "read access means." Because the read access means need only read from "a given register," and not from either register type, the corresponding structure is not – as Intel suggests – the multiplexing circuitry S1 and S2. As discussed above, multiplexers S1 and S2 are capable of reading from either register set. Instead, the specification discloses multiplexing circuitry 112 (shown in Figure 2A) for reading from Register Set RFB and multiplexing circuitry 150 (shown in Figure 3A) for reading from Register Set RA. (A17, Fig. 2A; A19, Fig. 3A; A31, 15:33-35, 15:47-53, 16:42-44). Multiplexers 112 and 150 are the corresponding structure in the specification that is used for retrieving data from ***a given register set***.

The Claimed "write access means." The parties agree that the structure corresponding to the claimed "write access means" is multiplexing circuitry 110 and 148 (in Figures 2 and 3). (A37, Fig. 2; A39, Fig. 3; *see* D.I.92, Tab A, pp.137-38). Intel's construction, however, includes

the same extraneous limitations as its construction for "writing means," *i.e.*, "at least the 4 multiplexers" and "controlled by instruction bit (DEST.) of figure 7." (D.I. 92, Tab A, p. 137). For the reasons discussed above, these additional limitations are improper.

6. Boolean Execution Unit Terms

Claim Language	Proposed Constructions
"Boolean execution unit" ('986 claim 11)	**Transmeta** – circuitry that executes instructions that generate Boolean results.
	Intel – a functional unit that is only used in performing bitwise logical combinations of Boolean register contents according to Boolean functions.
"Boolean combinational instructions each operating on one or more Boolean operands" ('986 claim 11)	**Transmeta** – instructions that perform Boolean operations on one or more Boolean values.
	Intel – instructions that perform Boolean operations on the results of previous Boolean operations.
"Boolean result" ('986 claim 11)	**Transmeta** – one or more bits representing logical "true" or "false" values.
	Intel – a single bit true/false indication from a Boolean function.

In addition to the integer register set, and the retypable integer/floating point register set, the Multiple Typed Register patents disclose the use of a third set of registers called "Boolean registers." These registers are used for holding Boolean results and are associated with a Boolean execution unit. Similar to the "execution unit" in claim 1 of the '449 patent (which was an agreed upon term between the parties, *see* D.I. 92, Tab A, p. 141), the Boolean execution unit is "circuitry that executes instructions that generate Boolean results." (D.I. 92, Tab A, p. 138).

Claim 11 of the '986 patent also specifies that the Boolean execution unit executes "Boolean combinational instructions each operating on one or more Boolean operands" (A54, claim 11). Transmeta proposes that this claim language means what it says — "instructions that perform Boolean operations on one or more Boolean values." (D.I. 92, Tab A, pp. 138-39). Intel, however, seeks to import an additional limitation by requiring the claimed "Boolean

operands" to result from "previous Boolean operations." (D.I. 92, Tab A, pp. 138-39). This contradicts the preferred embodiment, which provides that the Boolean execution unit also comprises a "numerical execution means for executing numerical comparison instructions to compare two multi-bit numerical operands." (A54, 19:36-39). Such comparisons are made on integers or floating point numbers, not Boolean values. (A28, 9:41-44). However, they do yield Boolean results, which in turn can be the operands in Boolean operations. (A29, 11:65-12:2). Therefore, Intel's "previous Boolean operations" requirement is too restrictive.

Finally, Intel seeks to limit the claimed "Boolean result" to "a single bit." (D.I. 92, Tab A, p. 139). Presumably, Intel bases this limitation on the embodiment in the specification where "each Boolean register is ***one bit wide***, indicating one Boolean value." (A27, 8:31-33) (emphasis added). But the specification also teaches that Boolean functions can "perform[] ***bitwise*** logical combinations of Boolean register contents." (A29, 12:5-8). A bitwise logical combination necessarily requires multiple bits. Thus, in one disclosed embodiment, the Boolean register is typically multiple bits wide, and the Boolean result of a bitwise logical combination of these Boolean registers would comprise multiple bits, not a single bit. (A32, 7:66-8:1).

C. The Register Renaming Patents[9]

The claims of the Register Renaming patents are directed to improving the speed of execution of out-of-order instructions in a processor. (A88, 1:44-47). Processors that can execute multiple instructions in parallel are called superscalar. Superscalar processors include the ability to issue and execute instructions out-of-order from the program order. (A88, 1:44-47; A994). Processors using superscalar techniques, therefore, are generally faster than those

[9] The Register Renaming patents are the '624, '526 and '433 patents. Although there are three separate patents in the family, the specifications are identical; only the claims differ. Therefore, the references herein are made to the '624 patent, with the understanding that they apply equally to the other two patents.

processors that can only execute instructions serially, *i.e.*, executing one instruction after another. A processor with a reduced instruction set (RISC) is an example of a processor with superscalar and "out-of-order" capabilities. (A88, 1:44-47). Processors with a complex instruction set (CISC) may also have superscalar capabilities, including the capability for out-of-order execution of instructions. (A992; A1004). Although the embodiments of the Register Renaming patent inventions are described with reference to a RISC processor, the invention and its applications are not limited for use in RISC processors. (A992).

The claimed inventions use a combination of three features to improve out-of-order execution in a processor: (1) data dependency checking; (2) register renaming; and (3) instruction scheduling. (A88, 1:47-49; A90-91, 6:18-7:16). Figure 1 of the specification illustrates the preferred embodiment. (A79, Fig. 1; A91, 7:26-27).

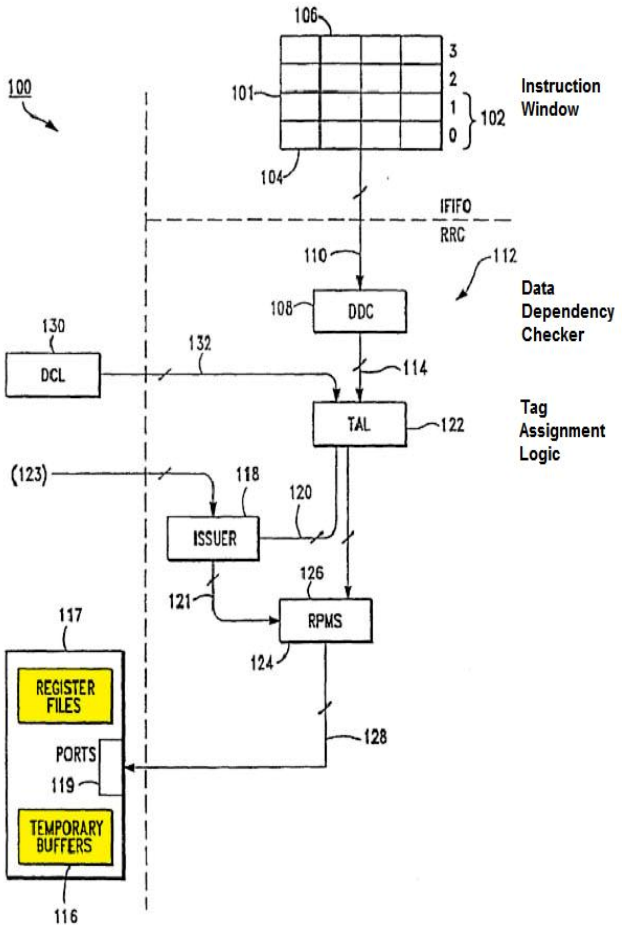

FIG.–1

To execute a program instruction, the processor fetches instructions for execution. Once decoded, the instructions are organized into a group. (A79, Fig. 1; A91, 7:63-67). The group of decoded instructions is known as an instruction window. (A79, Fig. 1; A91, 7:63-8:10). Instructions remain in the window until the processor resolves the dependencies between the instructions. (A90, 6:40-45; A91, 8:6-

10, 8:56-59). Data dependency checking is a process in which a device (data dependency checker) compares an instruction with another instruction and determines if there are any dependencies. (A90, 6:36-37). For example, a program might issue an instruction to add two numbers followed by an instruction to multiply the result of the previous instruction with a third number. The multiplication instruction is said to be dependent on the addition instruction because the multiplication instruction cannot be performed until after the addition instruction has completed. The result of the addition instruction can be considered an input required by the multiplication instruction. (*See* A90, 6:40-44).

Once dependencies are resolved, the results of instructions move from temporary buffers (A79, Fig. 1, 116) to an appropriate place in the register files (A79, Fig. 1, 117). This movement process is called "retirement." (A91, 8:59-61).

In the preferred embodiment, at least a portion of the instructions in the instruction window are checked for dependencies and then their results are assigned temporary storage locations in temporary buffers. (A91, 8:30-34, 42-46; A94, 13:6-10). If a dependency exists, the tag assign logic unit outputs a name for the temporary buffer location (also called a tag) and assigns that tag to the input required by the dependent instruction. (A90, 6:46-49, 55-63; A94, 13:6-7). The tags indicate the physical storage locations of the inputs for instructions. (A90, 6:62-63; A94, 13:7-10). Instructions access the contents of the temporary buffers by using the tags as a locator of the physical storage where the input data for each instruction is to be found. (A91, 8:15-19; A94, 13:44-62). The tags, therefore, rename the input register names identified in the instruction, and can name multiple physical registers for the same input register. (A94, 13:44-62; A460).

The instructions in the window can also be pre-assigned to locations in the temporary buffer in program order before being checked for dependencies. The instructions are then checked for dependencies and if a dependency exists, the dependency checker outputs the tag of the previously assigned temporary buffer location. (A96, claim 1).

1. "register renaming"

Claim Language	Proposed Constructions
"register renaming" ('624 claims 1, 13; '433 claims 1, 13)[10]	**Transmeta** – naming multiple physical registers for the same architectural register in order to reduce or eliminate storage conflicts.
	Intel – removing storage conflicts without actually renaming register addresses in the instruction.

The term "register renaming" appears in the preamble of the asserted claims of the '624 and '433 patents. For example, the preamble of '624, claim 1 states: "A system for register renaming in a computer system capable of out-of-order instruction execution, comprising:" (A96, claim 1).

Transmeta's proposed construction is the commonly understood meaning of register renaming. (B1098). Register renaming means "naming multiple physical registers for the same architectural register in order to reduce or eliminate storage conflicts."[11] (e.g., D.I. 92, Tab A, p. 144; *see also* B1098). Register renaming allows the duplication of registers by allowing one architectural register name to refer to data residing in multiple physical registers. (*See* A88, 6:11-18; A89, 3:19-30; A996). This meaning is set forth in the specification:

> Anti- and output dependencies are more properly called "storage conflicts" because reusing storage locations (including registers) causes instructions to interfere with one another even though conflicting instructions are otherwise

[10] "Register renaming" also appears in claims 7, 14, 15, 16 and 19 of the '624 patent, and claims 2, 3, 4, 6, 7, 14, 15, 16 and 19 of the '433 patent.

[11] An architectural register is a register name referred to by the program that runs on a microprocessor. (B1098).

> independent. Storage conflicts constrain instruction issue and reduce performance. But storage conflicts, like other resource conflicts, can be reduced or eliminated by ***duplicating the troublesome resource***.

(A88, 2:11-18) (emphasis added). The specification, therefore, explains register renaming as "duplicating the troublesome resource," – *i.e.*, naming additional physical registers with the same architectural register name so that instructions using that same architectural register can be executed in parallel. (*See* A88, 2:16-18; A89, 3:19-30, 52-56). The specification similarly explains that the same register identifier, *i.e.*, the same architectural register name, may be used to access different physical registers: "The same register identifier in several different instructions may access different hardware registers, depending on the locations of register references with respect to register assignments." (A89, 3:27-30; *see also* A89, 3:19-22). The specification is consistent with the common understanding of register renaming. (*see* A996; B1098).

Intel's proposed construction does not define what register renaming is. Instead, Intel adds a negative limitation to the preamble of the claim that prohibits the invention from "actually renaming register addresses in the instruction." (*See* D.I. 92, Tab A, p. 144). Contrary to Intel's proposed construction, the invention does ***not*** operate without renaming addresses "in the instruction." Rather, the specification teaches that:

> A Register Rename Circuit (RRC), which is part of the scheduling logic of the computer's IEU performs this function by locating dependencies between current instructions and then ***renaming the sources (inputs) of the instruction***.

(A91, 8:15-19) (emphasis added). The specification thus expressly states that, in the preferred embodiment, the sources (inputs) of the instruction (*i.e.*, architectural register names) are renamed in the instruction. (A91, 8:15-19). This is accomplished through the use of tags. This approach to register renaming is consistent with the claim language and was explained in the file history as follows: "[O]nce a dependent instruction is determined to be dependent on a previous

instruction, ***the operand of the dependent instruction is rewritten as the tag*** of the instruction on which it is dependent." (A460) (emphasis added).

2. "associating said index-addressable temporary storage location assigned to said previous instruction with said input" / "to associate said temporary storage location assigned to said previous instruction with said input"

Claim Language	Proposed Constructions
"associating said index-addressable temporary storage location assigned to said previous instruction with said input" / "to associate said temporary storage location assigned to said previous instruction with said input" ('526 claims 1, 5, 19, 34)	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this phrase means that the input is associated with the index-addressable storage location assigned to the dependent instruction.
	Intel: – [associating/to associate] the input from the dependent instruction with the temporary storage location assigned to the previous instruction without actually renaming the register address of the input in the dependent instruction.

The disputed claim language "associating said index-addressable temporary storage location assigned to said previous instruction with said input" appears in the asserted claims of the '526 patent (but not the '624 and '433 patents) (*see, e.g.*, A117, 17:29-31). This claim language should be given its plain meaning. The claim requires a temporary storage location (*i.e.*, a different physical register) to be associated with the input (*i.e.*, an architectural register name) of a dependent instruction. If the Court finds that a construction is necessary, the disputed language should be construed to mean that "the input is associated with the index-addressable storage location assigned to the dependent instruction." (D.I. 92, Tab A, p. 156).

Intel's proposed construction confirms that the claim needs no construction. Intel does not actually construe any of the words of the claim (*e.g.*, "associating," "temporary storage location," or "assigned to the previous instruction"). Instead, Intel reiterates that same exact language and then adds an extraneous limitation, *i.e.*, "without actually renaming the register address of the input in the dependent instruction." (D.I. 92, Tab A, p. 156). This limitation is

not required by the claim and is inconsistent with the specification's teaching that the input in the instruction is actually renamed in the instruction. (A91, 8:15-19).

3. "computer system" / "processor"

Claim Language	Proposed Constructions
"computer system" / "processor" ('624 claims 1, 7; '526 claims 1, 5, 19, 34; '433 claims 1, 7, 8, 9, 10, 12)	**Transmeta** – No construction necessary – plain and ordinary meaning.
	Intel – a reduced instruction set computer.

The terms "computer system" and "processor" are well-understood terms that do not need construction. (D.I. 92, Tab A, p. 144). Intel wants to import the preferred embodiment "reduced instruction set" or RISC into the asserted claims. (D.I. 92, Tab A, p. 144). The claims recite only that the computer system be "capable of out-of-order instruction execution." (*see,* '624 claim 1). None of the asserted claims require that the processor have a particular type of instruction set such as "reduced instruction set" or RISC. (*See, e.g.*, A96, 18:6-21).[12] Processors with a complex instruction set (CISC) are also capable of out-of-order execution. (A1004).

Describing the invention in its application to a RISC processor for ease of understanding, the '624 specification cites to a Johnson textbook, which states that "the properties of a RISC processor makes it easier to understand the application of superscalar techniques." (A88, 1:35-44; A992). And, "[m]any superscalar techniques apply equally well either to RISC or CISC architectures." (A992). Even those of skill in the art at Intel have admitted that the RISC/CISC distinction was blurring by 1994 because superscalar techniques are applicable to both RISC and CISC processors:

[12] The applicants obtained claims directed specifically to RISC processors in a related patent, the '499 patent, which has not been asserted in this action. The '499 patent issued from the parent of the application leading to the issuance of the '624 patent (and great-grandparent of the '433 patent). The '499 patent has system claims that expressly recite RISC (*e.g.*, A899, 17:55-18:9) and system claims that do not recite RISC (*e.g.*, A899, 18:18-37).

> This distinction between these two architectural approaches [RISC/CISC] is blurring – for example, the Pentium processor uses superscalar RISC technology to execute more than one instruction per clock cycle and branch prediction to speed instruction execution. In addition, with ever-increasing numbers of transistors available to circuit designers, many RISC processors are adding floating-point units and advanced memory-management operations previously found only on CISC processors. Ultimately, microprocessors are integrating the best of both technologies – cache technology, superpipelining, superscalar execution, and floating-point capability – to take microprocessors into the twenty-first century.

(B1010).

4. "instruction window"

Claim Language	Proposed Constructions
"instruction window" ('624 claims 1, 2, 7, 8, 13, 14; '433 claims 1, 7, 8, 13, 14) "instruction buffer" ('526 claims 1, 5, 13, 14, 15, 19, 26, 27, 28, 34)	**Transmeta** – a group of the instructions resulting from decoding that have not been retired.
	Intel – the group of instructions for which the computer system determines dependencies at the same time, wherein the number of instructions is equal in size to the number of storage locations in the temporary buffer.

Transmeta's proposed construction is the commonly understood meaning of "instruction window" and is supported by the intrinsic evidence. Intel's proposed construction, on the other hand, seeks to add unnecessary limitations from the preferred embodiments.

In a processor that performs out-of-order execution of instructions, the processor must examine instructions to find instructions that do not have dependencies and can be immediately executed. The processor decodes instructions, determines the fastest order in which the instructions can be executed and then "retires" instructions for which all dependencies have been resolved. The group of decoded instructions which have not yet been retired is referred to as an "instruction window." (A994).

With reference to Figure 1, the specification of the Register Renaming patents explains that the instruction window contains those instructions that have been decoded and are being examined by the Instruction Execution Unit:

> [Instruction Execution Unit] 100 can decode and schedule up to two buckets of instructions at one time. FIFO 101 stores 16 total instructions in four buckets labeled 0-3. ***IEU 100 looks at an instruction window 102***. In one embodiment of the present invention, window 102 comprises eight instructions.

(A79, Fig. 1; A91, 7:66-8:4; *see also* A90, 5:10-12, 5:25-28) (emphasis added). The specification further explains that "[o]nce the instructions in a bucket are executed and their results stored in the processor's register file, the bucket is flushed out a bottom 104 and a new bucket is dropped in at a top 106." (A91, 8:6-10; *see* A79, Fig. 1; *see also* A994). This process of moving instructions "to the appropriate place in register file 117" is called "retirement." (A91, 8:56-62). Once the instructions are retired, they are no longer a part of the instruction window. (A91, 8:2-10, 52-66). Therefore, as Transmeta proposes, the instruction window consists of the decoded instructions that have not yet been retired.

Intel's proposed claim constructions seek to unnecessarily add (1) a temporal limitation ("at the same time") and (2) a physical size limitation into the claimed instruction window. (D.I. 93, Tab A, pp. 144-45). The preferred embodiment includes eight instructions in the instruction window and allows all eight instructions to be checked for dependencies at the same time. (A91, 8:2-4, 15-19, 42-44). The specification, however, does not ***require*** that all eight instructions in the window be checked at the same time. To the contrary:

> This implementation of RRC 112 ***can*** check eight instructions at the same time, so a current instruction is defined as any one of those eight from window 102. ***It should become evident to those skilled in the art that the present invention can easily be adapted to check more or less instructions.***

(A91, 8:42-46) (emphasis added). Thus, although the size of the instruction window remains the same, the number of instructions in the window that are checked at the same time can be different. (A91, 8:42-46).

The claim language and specification also do not require the number of instructions in the instruction window to be equal in size to the number of storage locations in the temporary buffer. Intel's attempt to limit the claims in this regard is improper.

5. "data dependency checker"

Claim Language	Proposed Constructions
"data dependency checker" ('624 claims 1, 7; '526 claims 5, 15, 16, 19, 28, 29; '433 claims 1, 7)	**Transmeta** – logic that checks whether the input of an instruction is an output of a previous instruction in the instruction window.
	Intel – logic that compares the addresses of the inputs of each instruction in the instruction window to the address of the output of each previous instruction in the instruction window.

The parties' dispute regarding this term is closely related to the parties' dispute regarding "instruction window" discussed above. Transmeta's proposed construction is simple, easy to understand and consistent with the intrinsic evidence.

The specification explains that the purpose of the data dependency checker (DDC) is

> to locate the dependencies between the instructions ***for a group of instructions***. The DDC does this by comparing the addresses of the source registers of each instruction to the addresses of the destination registers of each previous instruction in the group.

(A90, 6:36-40) (emphasis added). The parties' dispute is whether the "group of instructions" that are checked for dependencies must include ***each*** instruction in the instruction window.

The specification teaches that: "[Data Dependency Checker] 108 determines where the input dependencies are ***between the current instructions***." (A92, 9:37-40) (emphasis added). In the preferred embodiment, the data dependency checker is capable of checking "eight instructions at the same time." (A91, 8:42-43). Therefore, in the preferred embodiment, "a current instruction is defined as any one of those eight from window 102," and the data dependency checker checks all eight. (A91, 8:42-44). Intel's proposed construction limits the

claims to this embodiment by requiring the data dependency checker to check for dependencies between "each instruction in the instruction window."

As discussed above, however, the specification also indicates, "It should become evident to those skilled in the art that the present invention ***can easily be adapted to check more or less instructions***." (A91, 8:44-46) (emphasis added). Accordingly, the specification expressly teaches that the data dependency checker is ***not*** required to check ***each*** instruction in the instruction window. Rather, the number of instructions within the window that are checked "can easily be adapted." (A91, 8:42-44).

6. "one of a plurality of storage locations being determined by a location of said instruction in an instruction window" / "storage locations determined by the location of an instruction in an instruction window"

Claim Language	Proposed Constructions
"one of a plurality of storage locations being determined by a location of said instruction in an instruction window" / "storage locations determined by the location of an instruction in an instruction window" ('624 claims 1, 7, 13)	**Transmeta** – storage locations are assigned based on the program order of instructions in the instruction window.
	Intel – each instruction in the instruction window maps to a specific, predetermined location in the temporary buffer based on that instruction's position in the instruction window.
"said one of said plurality of storage locations being assigned to said instruction in said instruction window" / "storage locations assigned to instructions in an instruction window" ('433 claims 1, 7, 13)	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this phrase means one of the plurality of storage locations in the temporary buffer is assigned to the instruction in the instruction window. **Intel** – each instruction in the instruction window maps to a specific, predetermined location in the temporary buffer based on that instruction's position in the instruction window.
"assigning a unique one of the plurality of index-addressable temporary storage locations to each one of said plurality of instructions in said instruction buffer" / "each one of said	**Transmeta** – a plurality of instructions in the instruction buffer are each assigned a single index-addressable temporary storage location.
	Intel – each instruction in the instruction buffer maps to a specific, predetermined temporary storage location based on that instruction's position in the instruction buffer.

plurality of instructions is assigned to a unique one of said plurality of temporary storage locations" ('526 claims 1, 5, 19, 34)	

Transmeta proposes that these terms be construed according to their plain and ordinary meaning. Intel's proposal is to not actually construe the words, but to add limitations to the claims, regardless of the claim language in question. The claim language does not require a mapping of each instruction in the instruction window to a specific, predetermined location in the temporary buffers "based on that instruction's position in the instruction buffer." (*See, e.g.*, A96, 18:6-21).

As stated in the Summary of the Invention, the invention only requires temporary buffers that temporarily store the instruction results. (A90, 6:25-26). In addition, the invention maps an instruction to a temporary buffer location. (A90, 6:49-51). There is no requirement that the location be based on the instruction's location in the buffer.

Intel's proposed claim construction deviates from the claim language and seeks to add multiple limitations. Neither the claim language nor the specification requires such limitations.

7. "means for passing"

Claim Language	Proposed Constructions
"means for passing said tags to read address ports of said temporary buffer" ('624 claims 6, 12)	**Transmeta** – The function performed by the claimed "means for passing" is passing said tags to read address ports of said temporary buffer for accessing said instruction execution results. The disclosed structure that corresponds to the function of the claimed "means for passing" is a bus. For example bus 128 shown in Figure 1.
	Intel – The claimed function is "passing said tags to read address ports of said temporary buffer for accessing said instruction execution results." The corresponding structure includes at least the Register File Port MUXes (RPM) 124 described at 17:45-58 and in Figure 4 of the '624 patent.

The parties agree that the claim element is a means-plus-function limitation that must be construed according to 35 U.S.C. §112, ¶6.

The Claimed Function of the "means for passing." The parties agree that the claimed function is "passing said tags to read address ports of said temporary buffer for accessing said temporary buffer for accessing said instruction execution results." (D.I. 92, Tab A, p. 148).

The Claimed "means for passing." The specification discloses bus[13] 128 as the structure that is used for passing tags to read address ports to connect to the temporary buffers for accessing the instruction execution results. (A79, Fig. 1; A96, 17:42-44). Intel's proposed construction is incorrect because it does not actually set forth the structure corresponding to the claimed function of "passing tags." Instead, Intel cites to the Register File Port MUXes (RPM) 124, shown in Figure 4, which is not necessary to perform the claimed function. *Micro Chemical*, 194 F.3d at 1258 (35 U.S.C. §112, ¶6 does not "permit incorporation of structure from the written description beyond that necessary to perform the claimed function."). RPM 124 is an unnecessary structural limitation because it is upstream of the structure that actually passes tags to the read address ports 119 of the temporary buffer 116. (A79, Fig. 1). The only structure of Figure 1 that passes tags to the read address ports of the temporary buffers is bus 128.

8. "tag assignment means"

Claim Language	Proposed Constructions
"tag assignment means for receiving data dependency results from a data dependency checker and for outputting a tag in place of a register address for an operand of a first instruction if said first instruction is	**Transmeta** – tag assignment logic for receiving data dependency results from a data dependency checker and for outputting a tag in place of a register address for an operand of a first instruction if said first instruction is dependent on a previous one of said plurality of instructions in said instruction window for said operand. "tag assignment means" is not subject to 35 U.S.C. § 112, ¶ 6 because it recites sufficient structure. If the court finds that

[13] A bus is a communication path between two points permitting data to be transferred. (*See* A95, 43-44).

dependent on a previous one of said plurality of instructions in said instruction window for said operand, wherein said tag represents an address of said operand in one of said plurality of storage locations. ('624 claim 7)	"tag assignment means for receiving . . . " is a means-plus-function limitation, it must be construed according to 35 U.S.C. §112, ¶ 6, as follows: The function performed by the claimed "tag assignment means" is receiving data dependency results from a data dependency checker and for outputting a tag in place of a register address for an operand of a first instruction if said first instruction is dependent on a previous one of said plurality of instructions in said instruction window. The disclosed structure that corresponds to the function of the claimed "tag assignment means" comprises instances of priority encoder 902 and mux 910 as shown in Figure 9.
	Intel – This term is subject to 35 U.S.C. § 112, ¶ 6. The claimed function is "receiving data dependency results from a data dependency checker" and "outputting a tag in place of a register address for an operand of a first instruction if said first instruction is dependent on a previous one of said plurality of instructions in said instruction window for said operand." The corresponding structure includes at least the Tag Assign Logic (TAL) 122 described at 14:55-15:33 and Figures 3 and 9 of the '624 patent.

Although this claim term uses the word "means," it has sufficient structure associated with it to define the structure claimed; *i.e.*, data dependency checker, instruction window, tag, register address, operand and instructions. Accordingly, it should not be construed as a means-plus-function element. If the Court nonetheless finds that § 112, ¶ 6 does apply, then Transmeta's proposed means plus function should be adopted because it follows the claim language and the specification. Intel omits the last portion of the function recited by the tag assignment logic: "if said first instruction is dependent on a previous one of said plurality of instructions in said instruction window." The specification discloses that the structure that corresponds to the function of the claimed "tag assignment means" comprises instances of priority encoder 902 and mux 910 as shown in Figure 9. (A87). Intel's recitation of the structure includes various structures shown in the specification that are unnecessary to perform the claimed function.

9. '526 method claims 1, 2 and 34: no order of steps required

Claims 1, 2 and 34 of the '526 patent are all method claims directed to methods of executing instructions that generally recite the steps of: (1) storing a plurality of instructions in an instruction buffer; (2) assigning a temporary storage location; (3) determining whether an instruction in the instruction buffer has a dependency and (4) associating the temporary storage location with the dependent instruction. (A117).

The logic of the steps recited by the claims requires only that the step of storing in an instruction buffer happen first and the step of associating the temporary storage location happen last. Whether the assignment of temporary storage locations occurs before or after the data dependency checking is of no consequence to the logic of the claim or the functioning of the invention. (*See* A90, 6:35-37, 58-63; A91, 8:15-19). These two steps are independent and it does not matter which one happens before the other. *Altirus, Inc. v. Symantec Corp.*, 318 F.3d 1363, 1369-71 (Fed. Cir. 2003) (The steps of a method claim do not have to be performed in the order written unless logic, grammar, or the content of the specification dictates otherwise).

D. The Speculative Address Translation Patents[14]

The Speculative Address Translation patents are directed to improved "address translation" in a computer, such as a microprocessor.[15] (A148, 1:7-8). When a computer

[14] As a threshold matter, Transmeta notes that Intel has insisted on submitting constructions for a total of 51 terms and phrases that, in Transmeta's view, need no construction. (*See* D.I. 92, Tab A, Sections II-V). Intel is unjustifiably attempting to import limitations into the claims, and unnecessarily complicates the claim construction process. Given the large number of terms that Intel construes, and the extended verbiage in Intel's proposed constructions, it is not feasible to even quote all of the terms and constructions in the confines of this brief, let alone discuss them. (Such terms are identified in other footnotes, *infra*. In addition, the terms "based on" [Term 7] and "corresponding portion of a complete translation of said second physical address" [Term 87] need no construction.).

[15] The Speculative Address Translation patents are the '503, '733, '668 and '699 patents. Although there are four separate patents in the family, the specifications are identical; only the

program reads data from, or writes data to, a computer memory, it uses an address to access the memory. The address is a number that allows the computer to go to the location in memory where the data is stored (for reading), or where the data is to be stored (for writing). (*E.g.*, B1006-08). As explained in more detail below, prior art address translation, particularly that used by Intel's microprocessors at the time, involved two translation steps: translating a "virtual address," used by software running on a microprocessor, into a "linear address," and then translating the "linear address" into a "physical address," which is the location in the actual memory used by the computer. (A150, 5:25-29). This two-step process was time consuming, and significantly reduced the performance of the system. (A149, 3:6-16).

The invention of the Speculative Address Translation patents speeds up this translation by using "fast," also called "speculative," addresses. In this approach, the prior art, two-step method of address translation still occurs. But whenever that happens, the system stores some information from that translation for later use. Then, when a subsequent address translation is performed, the system uses that stored information from a previous translation rather than calculating the new physical address from scratch. This speeds up the translation process. However, this faster address might be wrong — hence the term "speculative." The stored information from the prior translation may not be applicable to the current translation. In such event, the speculative address is not used, and the translation is done the slower, prior art way. However, computer systems store data such that a given virtual address is generally logically and physically close to previous virtual addresses, and thus the speculative addresses are usually correct. Thus, over the long run, this speculative address translation invention greatly improves performance. (A151, 7:5-19; A153, 12:42-63).

---

claims differ. Therefore, the references herein are made to the '503 patent, with the understanding that they apply equally to the other three patents.

1. Basic Memory Addressing Terms

The basic technical terms set forth below relate to address translation, are used throughout the claims, and are well understood by persons of ordinary skill in the art. The parties agree that it is appropriate to define these terms, but disagree on the definitions. Transmeta's definitions are based on the intrinsic evidence and better explain how address translation works in the context of the patents in suit.

Claim Language	Proposed Constructions
"physical address"[16] [17][Term 1]	**Transmeta** – a location in the computer's physical, *i.e.*, real, memory.
	Intel – an address that is sufficient to unambiguously specify the location of a desired unit of data equal in size to the smallest storage location addressable by the processor, typically one byte.
"virtual address" [Term 8]	**Transmeta** – an address in a segmented address space, the address having a segment identifier and a segment offset, and which is translated into a linear address if paging is enabled, and into a physical address if paging is disabled.
	Intel – a logical address having a fixed size and that translates into an intermediate "linear address" (as construed herein).
"linear address" [Term 19]	**Transmeta** – an address identifying a location in a continuous unsegmented address space, which is translated from a virtual address, and which is translated into a physical address.
	Intel – a logical address having a fixed size and that translates into an actual "physical address" (as construed herein).
"segmentation" [Term 13]	**Transmeta** – a form of memory management where a virtual address space is divided into segments, where each segment is allowed to start at any boundary, and have any length.
	Intel – the process of converting a "virtual address" (as construed herein) to a "linear address" (as construed herein).
"segment identifier" [Term 17]	**Transmeta** – the portion of a virtual address that identifies a segment in the virtual address space.

[16] The term, "physical memory addresses" [Term 70] is synonymous with "physical addresses." The terms "second physical address" [Terms 64, 86], and "third physical address" [Term 10] are self explanatory given the definition of "physical address." Intel's attempt to import lengthy additional limitations into these terms in the guise of construction should be rejected.

[17] References to "Term __" are to the claim terms so numbered in the Joint Claim Construction Chart, D.I. 92, Tab A, pp. 11-112.

Claim Language	Proposed Constructions
	Intel – the component of a "virtual address" (as construed herein) that uniquely identifies a variable-sized portion of data in a memory management system.
"segment offset" [Term 18]	**Transmeta** – the portion of a virtual address that identifies a location within a segment.
	Intel – the component of a "virtual address" (as construed herein) that is added to a base address to calculate a "linear address" (as construed herein).
"optional independent paging" [Term 14]	**Transmeta** – a form of memory management which can be enabled or disabled, and where the physical memory is divided into pages of predetermined size, which pages may be located independent of segment boundaries and lengths.
	Intel – the optional process of converting a "linear address" (as construed herein) to a "physical address" (as construed herein) following the completion of "segmentation" (as construed herein).
"page frame field"[18] *or* "page frame" [Term 2]	**Transmeta** – the portion of a physical address that identifies the physical location of a particular page. A "page" is a block of stored data of predetermined size.
	Intel – a portion of a "physical address" sufficient to unambiguously specify the location of a desired page of data.

Transmeta's definitions of these terms are based on the intrinsic evidence — the background art cited and discussed in the Speculative Address Translation patents. Specifically, the source of these terms comes from Intel's descriptions of its 1995-era microprocessors in existence as of the original filing date of the patents, such as Intel's Pentium microprocessor. (A153, 12:33-38). With one exception (*see* fn. 21), the patents use these terms in the same sense that Intel used them in this prior art context. The prior art is described in detail in Figure 1, the

[18] "Stored page frame" [Term 74] and "first page frame field" [Term 4], are self explanatory given the definition of "page frame." "Page offset portion" [Term 75] and "offset portions" [Term 69] refer to the page offset field, discussed in the text. Neither party has requested construction of "page offset field," a well understood term that specifies the location of data in a page. (A148, 2:4-6).

accompanying discussion in the Speculative Address Translation patents themselves, and various Intel references cited in the patents.[19]

"Address translation" arises in systems where different modes of addressing coexist. This is the case for the prior art Intel microprocessors discussed in the Speculative Address Translation patents. In the Intel microprocessors, a unit of data is stored in the physical memory at a "physical address," which is the actual location of the data in memory. (A148, 1:15-16). Programs running on Intel microprocessors use "virtual addresses" rather than physical addresses to locate data in memory. (A148, 1:12-15). Moreover, there are two types of memory organization that Intel uses for its microprocessors: (i) "segmentation;" and (ii) "optional independent paging." (A143 Fig. 1; A148, 2:63-66; A150, 5:1-3, 6:25-29; A861, 3:15-21). As a result, the virtual address is two steps away from the physical address — first, due to segmentation, the microprocessor must translate the virtual address into a "linear address." Then (unless paging is turned off) it must translate the linear address into the physical address. (A150, 5:25-29).

"Segmentation" divides the virtual address space into "segments." A segment can start at any location in memory and have any length (within the bounds of the memory). (A148, 2:47-49; A861, 3:46-51). Each segment has a "segment identifier" number that identifies the segment. The virtual address combines the segment identifier with a "segment offset" number that locates data within the segment. (A148, 1:47-58). Thus, the virtual address specifies the location of a segment and the location of the data in the segment.

[19] *E.g.*, A143, Fig. 1; A148, 1:42-3:5; A149-50, 4:55-6:29; A853-54, Figs. 2-3; A860-62, 1:18-5:63; A948-50, Figs. 1-3; A960-61, 1:17-4:58; B11-22. For additional background, Intel references that describe this technology are at B1006-08, B1011-1015 and B1043-71.

To translate the virtual address into a linear address, a "segmentation unit" takes the segment identifier and uses it to look up the "base" of the segment in a segment descriptor register.[20] The base is the starting location in memory of the segment. The base is added to the segment offset to get the "linear address." (A148, 1:58-62; A150, 5:30-44). A linear address is "continuous" and "unsegmented." (B1044, p. 3-1). Figure A shows pertinent portions of Figure 1 which demonstrate this translation from virtual to linear address (A143):

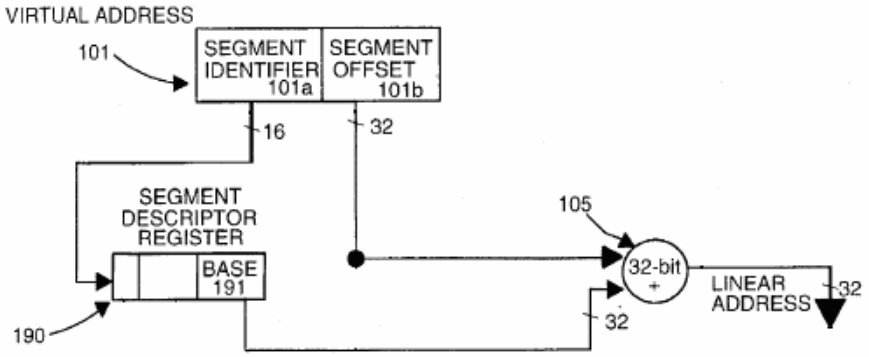

Figure A

As mentioned above, the second stage of address translation in the 1995-era Intel microprocessors is optional independent paging. "Paging" divides the memory into pages which, unlike segments, all start at fixed boundaries and have the same size. (A148, 1:63-66). "Independent" paging refers to the fact that the size and positions of the pages are independent of the size and positions of the segments — the two memory division schemes exist independently of each other. (A148, 2:37-52).[21] "Optional" paging means that the paging can be turned on or off in order to allow a paging-capable microprocessor to function with an older operating system that does not handle paging. (A148, 2:53-59). If paging is turned off, the above-mentioned

[20] Alternatively, the segment lookup could use a table stored in memory (A150, 5:7-10).

[21] In the prior art, "independent" also referred to the fact that address information stored in the segmentation unit was physically and logically separate from that in the paging unit. The Speculative Address Translation patents do not have that requirement. (A150, 6:8-14, 6:46-49).

linear address is the same as the actual physical address in memory, and no further translation is needed. (A150, 5:25-29, 5:54-56).

If paging is turned on, the linear address is translated into the physical address by the "paging unit." The linear address has at least two "fields": a "page number" field and a "page offset" field.[22] The page number is used to look up the "page frame field" (or "page frame," for short) of the page in a "page cache."[23] The page frame field locates the beginning of the page in physical memory. The page offset field locates the data within the page. The page frame field and page offset field are combined to form the final physical address of the data. (A148, 1:63-2:15; A150, 5:56-6:7). Figure B shows the pertinent portions of Figure 1 that demonstrate this translation from linear to physical address. (A143).

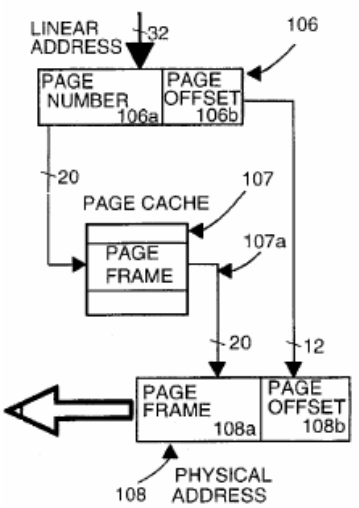

Figure B

In sum, Transmeta's constructions of these basic terms are taken directly from the specifications of the patents in suit and other intrinsic evidence. Intel's constructions do not comport with this evidence, and instead either impose unwarranted additional limitations into the terms, or incompletely define them. For example, Intel's definition of "optional independent paging" ignores "independent" and erroneously suggests that paging could be omitted from the system entirely, as opposed to optionally being enabled or disabled.

[22] Some systems used linear addresses that also have a "page directory" field (A150, 5:63-67).

[23] Alternatively, the page number lookup could use a table stored in memory (A150, 5:18-24).

An additional claim construction issue concerns the fact that the terms "segmentation" and "optional independent paging" appear in the preamble of many of the claims in suit.[24] A preamble term is a claim limitation if it is "necessary to give life, meaning, and vitality" to the claim. *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1306 (Fed. Cir. 1999). Each of the claims in question refers in the body of the claim to a "linear address." As discussed above, linear addressing is inextricably intertwined with segmentation and optional independent paging. It makes no sense to refer to the latter without the former. For at least this reason, these preamble terms are properly considered claim limitations.

2. "Fast/Tentative/Speculative Addressing," And Related Terms

There are several groups of terms that all relate to the improved speculative address translation approach of the patents:

Claim Language	Proposed Constructions
"fast physical address"[25] [Term 20] "tentative physical address" [Term 36] "speculative physical address" [Term 45] "speculative memory address" [Term 80]	**Transmeta** – an address specifying the location of data that may or may not be the desired location, and which is available sooner than an actual physical address.
	Intel – an address sufficient to unambiguously specify the location of a unit of data equal in size to the smallest storage location addressable by the processor that may or may not be the desired unit of data, and which is generated quicker than a "physical address" (as construed herein).

The parties agree that the terms "fast memory access," "tentative memory access," and "speculative memory access" have the same meaning, and all connote "fast" in the sense of speed as well as the speculative or tentative nature of the address. These terms all refer to the faster, speculative address translation improvement over the prior art address translation

[24] '733 claims 1, 28, 36, 39, 48, 57, 63, 69; '688 claims 1, 15, 20; '699 claims 1, 7, 10, 13.

[25] The term "fast page offset" [Term 30] refers to the page offset of a fast physical address. The term "speculative physical page frame address" [Term 67] refers to the page frame field of a speculative physical address. Neither term needs to be construed.

approach described above. The speculative address is in part based on information saved from a previous prior-art-type address translation, and is generated faster than the prior art process. The fast, speculative address may be wrong, but it is correct often enough so that the overall effect is a significant performance improvement. As the Abstract of the '733 patent states (A158):

> A fast physical address is generated in parallel with a fully computed virtual-linear-physical address in a system using segmentation and optional paging. This fast physical address is used for a tentative or speculative memory reference, which reference can be canceled in the event the fast physical address [is incorrect].

Although there is some agreement on these terms, Transmeta's definition is clear and to the point, in contrast to Intel's wordy and confusing definitions.

Claim Language	Proposed Constructions
"fast memory reference"[26] [Term 37] "fast memory access" [Term 77] "tentative memory access" [Term 42] "speculative memory access" [Term 46]	**Transmeta** – using the fast/tentative/speculative physical address to locate data in memory.
	Intel – using a "fast/tentative/speculative physical address" (as construed herein) to refer to memory in the same manner as if the final fully translated full "physical address" (as construed herein) was already available.

A "fast memory reference" and like terms refer to the use of the fast physical address to locate data in memory. Transmeta's definition is straightforward. In contrast, Intel attempts to

[26] The terms "memory reference . . . based on the fast physical address" [Term 32] and "memory reference using the speculative physical address" [Term 79] have the same meaning as "fast memory reference" and "speculative memory reference." Intel construes the terms "memory access request based on said third physical address" [Term 11], "generating a memory access" [Term 21], and "memory access . . . using said second . . . physical address" [Term 61] as having the exact same meaning as "fast memory reference." However, these claim terms mean what they say, and need no further construction. Intel is trying to read specific aspects of the preferred embodiment into these more broadly worded elements.

insert a requirement that the memory access be "in the same manner" as for the non-speculative address. There is no justification for such a limitation.

Claim Language	Proposed Constructions
"partial linear address information relating to said virtual address" [Term 24] [also Terms 33, 38, 40, 48, 57, 83]	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means a portion of a linear address translated from the virtual address referred to in claim 1.
	Intel – a portion of a "physical address" (as construed herein) sufficient to unambiguously specify the location of a byte of data within a page and that is used together with a "fast page frame" (as construed herein) to form a "fast physical address" (as construed herein).
"physical address information from a different virtual address" [Term 23] [also Terms 12, 34, 39, 41, 43, 49, 50, 58, 73, 78, 81, 85, 88]	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means at least a portion of a physical address translated from a prior virtual address.
	Intel – a "page frame field" (as construed herein) that may or may not specify the location of the desired page of data and that is obtained from the "physical address" (as construed herein) used in the previous request for data from the segment from which data is currently being requested.

Neither the phrase "partial linear address information relating to said virtual address," nor the phrase "physical address information from a different virtual address," nor the other similar phrases referenced above, need to be construed. Intel is comparing this claim language to the preferred embodiment, and improperly reading details of that embodiment into the broader language of the claims. Specifically, in the preferred embodiment, the fast, or speculative, address is created by taking the page frame field from a prior address translation and combining it with the page offset of the current linear address being translated. (A146, Fig. 3B, element 303; A153, 11:35-40). Although that is how the claim language in question reads on the preferred embodiment, there is no reason to require that the claim be construed such that, for example, "partial linear address information" ***must*** "unambiguously specify the location of a

byte of data," or that "physical address information" ***must*** be a "page frame field," as Intel would require.

Claim Language	Proposed Constructions
"information from the first address translation" [Term 47] [also Terms 47, 51, 52, 60, 63]	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means at least a portion of an address translated during the first address translation.
	Intel – a "page frame field" (as construed herein) that may or may not specify the location of the desired page of data and that is obtained from the "physical address" (as construed herein) used in the previous request for data from the segment from which data is currently being requested.

Again, this phrase, and the related phrases referenced above, need no construction. As with "physical address information from a different virtual address," Intel would restrict the phrase "information from the first address translation" to a "page frame field" obtained from the first address translation. Although this is the case for the preferred embodiment, there is no justification for so narrowing this language as a matter of claim construction.

Claim Language	Proposed Constructions
"**combination/combined/combining**" [Term 22]	**Transmeta** – association/associated/associating.
	Intel – "the fast page offset" (as construed herein) is concatenated with a "fast page frame" (as construed herein) to generate a "fast physical address" (as construed herein).

In the context of the claims in which "combination" or its variants are used, the plain ordinary meaning of combine in the sense of "associate" is the appropriate definition. (*E.g.*, B1076). Intel's attempt to import additional functional requirements onto these ordinary English words is unwarranted.

Claim Language	Proposed Constructions
"actual physical address"[27] [Term 16]	**Transmeta** – a non-speculative physical address.
"calculated physical address" [Term 25] "computed physical address" [Term 44]	**Intel** – an address that is sufficient to unambiguously specify the location of a desired unit of data equal in size to the smallest storage location addressable by the processor, typically one byte.

The parties agree that "actual physical address," "calculated physical address" and "computed physical address" have the same meaning, and are the addresses that are determined in accord with the prior art two-step translation method discussed above. In other words, these addresses are non-speculative. Again, Transmeta's definition is clear and to the point.

3. Memory And Storage Terms

Claim Language	Proposed Constructions
"address translation memory" [Term 72] [also Terms 71, 82, 84]	**Transmeta** – No construction necessary – plain and ordinary meaning.
	Intel – a memory for storing the page frame associated with the current memory request that is indexed by virtual address information so that it can be rapidly accessed to generate a "fast physical address" (as construed herein) in response to the next request for data in the segment from which data is currently being requested.
"storing/stored" [Term 9]	**Transmeta** – No construction necessary – plain and ordinary meaning. If the Court decides a construction is necessary, this term means "placing/placed in storage."
	Intel – storing the page frame for the current memory request in a memory that indexed by virtual address information so that it can be rapidly accessed to generate a "fast physical address" (as construed herein) in response to the next request for data in the segment from which data is currently being requested.

[27] The terms "calculated page frame" [Term 26] and "actual physical page frame address" [Term 65] simply refer to the page frame fields of the calculated or actual physical addresses, and need no construction.

Intel has attempted to construe the above-referenced terms directed to "memory" and "storage" in memory in a way that limits the claims to an aspect of the preferred embodiment that is not required by the claim language. In particular, as discussed above, in the preferred embodiment, the fast (speculative) address is based in part on the page frame field from a prior address translation, called the "Last Page Frame." (A146, Fig. 3B, element 397). In the preferred embodiment, the Last Page Frame is stored in the Segment Descriptor Memory. (A146, Fig. 3B, element 390). It is true that, in the preferred embodiment, this memory is indexed by virtual address information. (A153, 11:11-18). However, there is nothing in the claims that require the memory to be indexed via the virtual address information. Intel is again attempting to read limitations into the claims that are not there.

Also, just before the Joint Claim Construction Chart was due, Intel inserted a footnote into the chart that stated (D.I. 92, Tab A, p. 11):[28]

> Intel's position is that each of the asserted claims of the '503, '733, '668 and '699 patents should be construed to require the use of a memory for storing previously generated "page frame fields" (as construed herein) that is indexed on the basis of virtual address information only.

Thus, in addition to the above-referenced memory terms, Intel now bootstraps this same argument into ***all*** of the claims of the Speculative Address Translation patents. Again, although unasserted claims do impose this limitation (*e.g.*, '733 claim 23), none of the asserted claims are so limited.

E. The Intel Patents

1. The '375 Patent

The '375 patent is directed to a power control circuit and method for reducing power consumption of an electronic device. The circuit of the '375 patent allegedly improves upon the

[28] Transmeta objects to Intel's omnibus construction as untimely.

prior art by scaling voltage and frequency supplied to the electronic device if the temperature of the device has exceeded a thermal band. Unlike the invention of Transmeta's '061 patent, the '375 patent circuit does not permit the processor to execute instructions while the voltage is changing.

The power control circuit disclosed in the '375 patent controls power consumption of a microprocessor, controller or other electronic device. Reproduced below is FIG. 3 of the '375 patent, which shows a block diagram of the power control circuit of the alleged invention and an electronic device, CPU 110. (D3). The Clock Generation Circuit portion (160) supplies a scalable frequency to the electronic device (Central Processing Unit, "CPU" 110) via clock line (161) and the Power Supply Circuit portion (170) provides a scalable voltage to the electronic device (CPU 110) via power line (173).

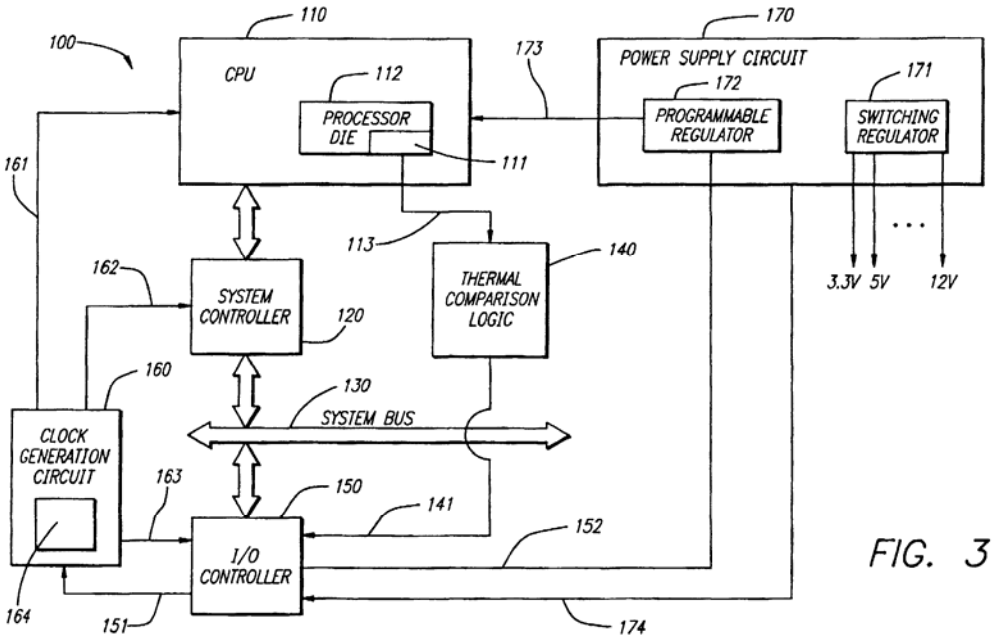

FIG. 3

(a) "electronic device"

Claim Language	Proposed Constructions
"electronic device" ('375: claims 1, 2-4, 6-7, 17, 18, 30, 30-31, and 33)	**Transmeta** – a self-contained electronic component.
	Intel – No construction necessary.

"Electronic device" is not explicitly defined in the '375 patent. The patent refers only to examples of electronic devices – namely, microprocessors and controllers. (D6, 1:28-31, 2:5-10; D7, 4:3-11). Transmeta's proposed claim construction finds support in the specification, which, as shown above, teaches that the electronic device is a distinct and separate component from the power supply circuit and the clock generation circuit – which are described and claimed as supplying voltage and clock signals ***to*** the electronic device.

(b) "supplies...to" and "provides...to"

Claim Language	Proposed Constructions
"supplies...to" '375 (claims 1, 16, 30, and 33)	**Transmeta** – supplies to, from an external source.
	Intel – supplies a clock signal having a frequency that can be increased or decreased as necessary to.
"provides...to" '375 (claims 1, 16, 30, and 33)	**Transmeta** – provides to, from an external source.
	Intel – provides a power supply signal having a voltage that can be increased or decreased as necessary to.

The phrases "supplies...to" and "provides...to" appear in the claims in connection with the clock generation circuit and the power supply circuit and state that these circuits supply/provide a signal ***to*** the electronic device. Therefore, for similar reasons to those outlined above for the term "electronic device," the phrases should be defined as supplying/providing to, from a source external to the electronic device.

(c) "event" and "said event"

Claim Language	Proposed Constructions
"event" '375 (claims 1 and 16)	**Transmeta** – a temperature related occurrence.
	Intel – circumstances warranting a change in power consumption.
"said event" '375 (claims 2, 4, and 17-18)	**Transmeta** – the same temperature related occurrence.
	Intel – circumstances warranting a change in power consumption.

The terms "event" and "said event" should be defined to be temperature related occurrences. Those are the only constructions supported by the claims. Claim 2 recites the "power control circuit according to claim 1 further comprising a thermal detection circuit that monitors a temperature...and outputs a third signal...upon detecting said event." (D9). The antecedent basis of the term "said event" is the term "event" in claim 1. Therefore, because the detection circuit which detects "said event" in claim 2 is a temperature monitoring circuit, the "event" in claim 1 necessarily is a temperature related occurrence. This claim construction is further supported by language in other claims, including claim 7, which depends from claim 1 and states that, in addition to the "event," the claimed circuit ***further*** detects percentage of idle time. (D9). Temperature and idle time are the only two operating characteristics disclosed in the specification. The claims treat them separately, with temperature being the "event" of claims 1, 2, 4, and 16-18. (D9).

(d) "thermal band" ('375: claims 3-4, 18-19, and 33)

Claim Language	Proposed Constructions
"thermal band" '375 (claims 3-4, 18-19, and 33)	**Transmeta** – operating temperature range defined by upper (maximum) and lower (minimum) temperature limits.
	Intel – acceptable operating temperature range defined by upper (maximum) and lower (minimum) temperature limits.

The parties generally agree upon this construction, except that Intel inserts the limitation "acceptable." The term "acceptable" is not known to have a particular meaning in the art and does not appear in the specification or the file history of the '375 patent. There is no valid reason to insert the word "acceptable" into this construction.

2. The Address Translation Patents[29]

Microprocessors typically include main memory and secondary memory. Main memory (*e.g.*, Random Access Memory) is much faster than secondary memory (*e.g.*, a hard disk drive), but also more expensive. Virtual memory allows microprocessors to extend main memory space into secondary memory space. Virtual memory microprocessors use techniques well known in the art (*e.g.*, paging or segmentation, or both) to translate virtual memory addresses into physical addresses and thereby simulate a larger main memory.

The '554 and '605 patents are directed to physical address size selection and page size selection in an address translator. These patents allegedly improve upon the prior art by allowing a larger physical address to be addressed and more than one page frame size to be selected during address translation while retaining compatibility with previous architectures. The patents teach that backward compatibility is maintained via the use of bits stored in the control unit in the microprocessor – these bits indicate when the paging unit within the microprocessor must support more than one page frame size and when the processor must support translation to a physical address that is larger than the linear address.[30]

(a) "control unit" and "paging unit"

Claim Language	Proposed Constructions
"control unit" '554 (claims 1 and 19) '605 (claim 1)	**Transmeta** – control circuitry including registers within the microprocessor.
	Intel – No construction necessary.

[29] The Address Translation patents are the '554 and '605 patents. Although there are two patents in the family, the specifications are identical; only the claims differ. Therefore, the references herein are made to the '554 patent, with the understanding that they apply equally to the '605 patent.

[30] For an explanation of the terms "linear address" and "physical address," see *supra* pp. 38-41.

"paging unit" '554 (claims 1 and 19) '605 (claims 1 and 2)	**Transmeta** – circuitry within the microprocessor used in paging.
	Intel – a unit to perform "paging" (as construed herein).

Transmeta's proposed claim constructions are supported by the claims, which state that the paging unit is "coupled to said control unit." This language prescribes a physical relationship between the two units, defining them to be hardware.

Transmeta's constructions are also consistent with the specification and the cited references. The '554 patent specification discloses that "[t]he ***microprocessor*** 100 also ***includes a page unit*** 110 ***which includes*** logic and additional ***circuitry*** including hardware, PLA programming sequencing control, and pointer registers 112."[31] (D30, 11:47-50; *see also* D22, Fig. 10; D820-821, Figs. 2, 3). The '554 patent also teaches that the "microprocessor includes a control unit 104 which has numerous registers provided therein..." (D30, 11:45-46; *see also* D22, Fig. 10; D255). This definition of the control unit and paging unit in the specification further support Transmeta's constructions.

Intel's construction of the term "paging unit" is overly broad, ambiguous, and inconsistent in that it only states the function of the element, not what a "paging unit" is.

(b) "page frame size"

Claim Language	Proposed Constructions
"page frame size" '554 (claims 1-3 and 19-21) '605 (claims 2-3)	**Transmeta** – the size of a contiguous aligned block of physical memory.
	Intel – fixed size of the memory unit used in "paging" (as construed herein).

Transmeta's proposed claim construction is supported by both the specification and the cited references. The specification of the '554 patent states that "the 'page frame' is the physical

[31] As discussed *supra* p. 9, registers provide physical storage locations in the processor.

memory itself." (D25, 2:8-9). Intel's own references further elaborate on this definition, characterizing a page frame as "a unit of contiguous addresses of physical main memory." (D503; D514).

Intel's construction introduces ambiguity into the claim term, because it refers to "the memory unit" yet provides no explanation of what that memory unit is.

3. The Pack/Unpack Patents , Multiply-Add Patent and Intra-Add Patent[32]

Microprocessors often need to manipulate large amounts of data which may be represented by only a small number of bits. The number of those bits is often much less than the size of the data bus. For example, audio data typically requires 8 or 16 bits, but the data bus or storage register may be 32 or 64 bits wide. To improve efficiency, prior art processors used packed data formats to increase the capacity of data being processed. Packed data allows all available bits of the data bus to be utilized by completely filling the storage unit with multiple data elements, e.g., four data elements of 8 bits each for a 32 bit wide bus or register. Three of the Intel patent "families" – Pack/Unpack, Multiply-Add, and Intra-Add – are directed to data manipulation of such packed data operands.

The Pack/Unpack patents teach an instruction that specifies performing a pack or unpack operation on multiple data elements in parallel. These patents allegedly improve upon the prior art by using a single instruction to pack or unpack multiple packed data elements. The pack operation copies a part of each data element from a first source and a part of each data element from a second source into a destination as a plurality of packed separate data elements. The

[32] The Pack/Unpack patents are the '101 and '275 patents. Although there are two patents in the family, the specifications are identical; only the claims differ. Therefore, the references herein are made to the '275 patent, with the understanding that they apply equally to the '101 patent. The Multiply-Add patent is the '634 patent. The Intra-Add patent is the '529 patent.

unpack operation copies less than all data elements from first and second packed sources into a destination as a plurality of separate data elements.

The Multiply-Add patent teaches a method of performing a multiply-add operation on packed data operands. In one embodiment, two multiply-add operations are performed using a single multiply-add instruction (*see* D126, Tables 3a & b). First, pairs of data elements from two packed data operands are multiplied together to generate four products. Then, those intermediate results are summed by pairs and stored as two separate data elements in a third packed data register.

The Intra-Add patent teaches a method of performing an intra-add or "horizontal add" operation on packed data. The intra-add operation adds together pairs of data elements in a packed data operand. One embodiment of the invention (D138, Fig. 2) showing packed data elements from two source operands being added and stored as data elements of a resulting packed data is reproduced below:

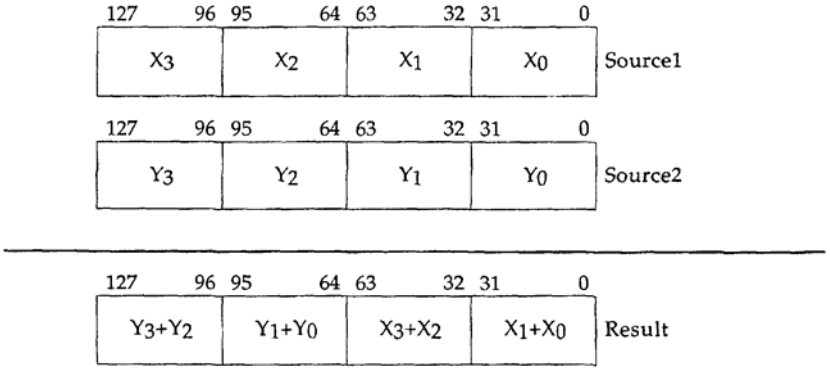

FIGURE 2

(a) "packed data"

Claim Language	Proposed Constructions
"packed data" '101 claims 1, 6, 9, 14; '275 claims	**Transmeta** – unit of data that is fully populated with a plurality of data elements of the same size.

1, 4-6, 9-10; '634 claims 3-6, 10, 12, 14; '529 claims 30-31, 33-34, 36, 38, 48	**Intel** – unit of data that consists of a plurality of data elements of the same size.

The term "packed data" should be construed as a "unit of data that is fully populated with a plurality of data elements of the same size." The unit of data is completely filled up – all bits are being used to store data.

The specifications of the Pack/Unpack, Multiply-Add and Intra-Add patent families illustrate a variety of packed data formats: packed byte, packed word, and packed doubleword. Reproduced below is Fig. 5a of the '275 patent[33] that shows the different formats of packed data:

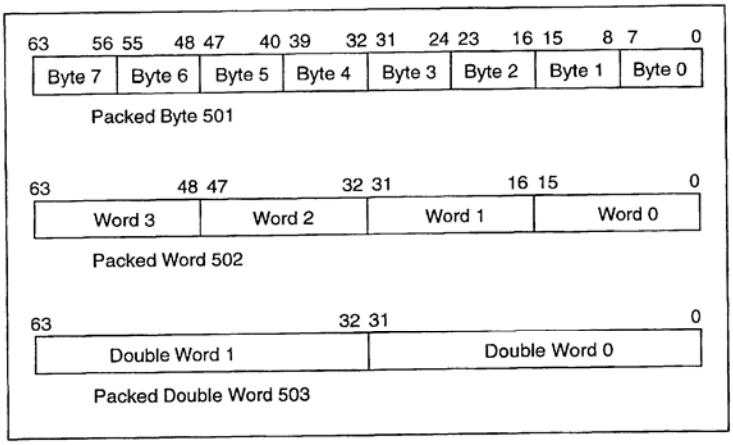

Figure 5a

As shown and provided in the specification, for each type of packed data, "all available bits are used in the register." (D108, 7:42-43; D128, 10:40). By making sure no bits are wasted, the packed data "storage arrangement increases the storage efficiency of the processor." (D108, 7:43-44; D128, 10:40-42). Transmeta's proposed claim construction encapsulates this concept of storage efficiency from the specifications of all four patents.

[33] Fig. 5a of the '275 patent (D95) is inherently identical to Fig. 4 of the '634 patent (D119) and Fig. 6 of the '529 patent (D147).

Intel's proposed claim construction is overly broad and contradicts the plain language of the specification. Under Intel's construction, all available bits need not be used, and it would not necessarily follow that storage efficiency would be increased.

(b) "decoding"

Claim Language	Proposed Constructions
"decoding" '101 claims 1, 9; '275 claims 1, 6; '529 claims 30, 33	**Transmeta** – transforming an external representation of an instruction into internal operations or commands.
	Intel – No construction necessary.

Transmeta's proposed construction is supported by the intrinsic evidence. The Pack/Unpack patents describe a decoder that "decodes...the operation code for the appropriate...operation." (D110, 11:10-11). The Intra-Add patent explains that the "[d]ecoder ... is used for decoding instructions received by processor 110 into control signals and/or microcode entry points." (D153, 6:38-40). In other words, the patents teach that "decoding" involves taking the form of the instruction that is external to the decoder and transforming it to enable the functional/execution unit to perform the appropriate internal operations or commands. (D110, 11:10-19, 11:24-29; D148, Fig. 8; D153, 6:38-44).

(c) "copying"

Claim Language	Proposed Constructions
"copying" '101 claims 1, 9; '275 claims 1-2, 6-7	**Transmeta** – bitwise replication independent of the value of the data.
	Intel – No construction necessary.

The term "copying" appears in claims of both the '275 and '101 patents and should therefore be construed consistently for both patents. Transmeta's proposed claim construction applies to both patents and is supported by the specification and the prosecution history. Figure

7 and its description teach that all of the bits are replicated even if the result exceeds the maximum possible value of the data. (D100, Fig. 7; D110, 11:61-12:24).

(d) "each include[s] either two, four, or eight data elements"

Claim Language	Proposed Constructions
"each include[s] either two, four, or eight data elements" '101 claims 4, 12; '275 claims 3, 8	**Transmeta** – the computer system has the capability of manipulating each of the specified alternatives.
	Intel – No construction necessary.

The phrase "each include[s] ***either*** two, four, ***or*** eight data elements" in the claim term should be construed to mean that the computer system is able to perform all of the listed options. Transmeta's proposed construction is supported by the specification. In describing the pack and unpack operations, the Pack/Unpack patents teach that the functional unit is enabled to operate on pluralities of all of two, four or eight data elements. (D103, Fig. 9).

(e) "intermediate data elements," "intermediate result data elements," and intermediate result"

Claim Language	Proposed Constructions
"intermediate data elements" "intermediate result data elements" "intermediate result" ('634 claims 5, 10, 12)	**Transmeta** – a result that is a fully calculated product.
	Intel – No construction necessary.

The Multiply-Add patent teaches that intermediate results are generated during the multiply-add operation.[34] These intermediate results are fully calculated products generated by multiplying together data elements from two packed data sources. The patent discloses that these intermediate results are "summed by pairs" to produce the result for the multiply-add instruction. (D126, 5:56-61; D129, 12:51-63). In order for that summation by pairs to generate a

[34] The '634 patent refers to these intermediate results as either "intermediate data elements," "intermediate result data elements," or "intermediate result[s]." These different terms are used interchangeably and have the same meaning throughout the patent.

final result, the intermediate results must themselves already be fully calculated products. Thus, Transmeta's construction is consistent with and finds support in the specification.

V. CONCLUSION

For the foregoing reasons, Transmeta respectfully requests that its proposed claim constructions be adopted by the Court.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Karen Jacobs Louden (#2881)

Jack B. Blumenfeld (#1014)
Karen Jacobs Louden (#2881)
Richard J. Bauer (#4828)
klouden@mnat.com
1201 N. Market Street
Wilmington, DE 19899
(302) 658-9200
Attorneys for Plaintiff Transmeta Corporation

OF COUNSEL:
Robert C. Morgan
ROPES & GRAY LLP
1211 Avenue of the Americas
New York, NY 10036-8704
(212) 596-9000

Norman H. Beamer
Sasha G. Rao
Gabrielle Higgins
ROPES & GRAY LLP
525 University Avenue
Palo Alto, CA 94301
(650) 617-4000

John O'Hara Horsley
TRANSMETA CORPORATION
3990 Freedom Circle
Santa Clara, CA 95054

October 19, 2007

1284853

CERTIFICATE OF SERVICE

I, the undersigned, hereby certify that on October 19, 2007, I electronically filed the foregoing with the Clerk of the Court using CM/ECF, which will send notification of such filing(s) to the following:

Josy W. Ingersoll

I also certify that copies were caused to be served on October 19, 2007, upon the following in the manner indicated:

BY HAND

Josy W. Ingersoll
John W. Shaw
YOUNG, CONAWAY, STARGATT & TAYLOR, LLP
The Brandywine Building
1000 West Street, 17th Floor
Wilmington, DE 19801

BY EMAIL

Steven S. Cherensky
Jessica L. Davis
WEIL, GOTSHAL & MANGES LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065

/s/ Karen Jacobs Louden (#2881)
_____
Karen Jacobs Louden (#2881)
klouden@mnat.com